Fuzzy Genetic Algorithm For VLSI Floorplan Design

by

Hakim Adiche

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

November, 1997

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

FUZZY GENETIC ALGORITHM FOR VLSI FLOORPLAN DESIGN

BY

Hakim ADICHE

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER ENGINEERING

November 1997

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN, SAUDI ARABIA

COLLEGE OF GRADUATE STUDIES

This thesis, written by

HAKIM SALAH ADICHE

under the direction of his Thesis Advisor and approved by his Thesis Committee, has been presented to and accepted by the Dean of the College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER ENGINEERING

Thesis Committee

Dr. Habib Youssef (Chairman)

Dr. Sadiq M.Sait (Co – Chairman)

Dr. Mostafa Abd – El – Barr (Member)

Dr. Mohammed Jamoussi (Member)

Dr. Khalid M. Al – Tawil
(Department Chairman)

Dr. Abdallah M. Al – Shehri
(Dean, College of Graduate Studies)

7-12-97
Date

Acknowledgements

All praises are for ALLAH *subhanahu-wa-ta-Aaala*, the Most Compassionate, the Most Merciful. May peace and blessings be upon his Prophet Muhammad, and his family. I thank Almighty Allah for the completion of this work. I acknowledge the support and facilities provided by the King Fahd University of Petroleum and Minerals.

I would like to express my profound gratitude and appreciation to my thesis committee chairman Dr. Habib Youssef, for his constant help, guidance and the attention and efforts that he devoted to complete this work. I would like to thank co-chairman Dr. Sadiq Sait, whose continuous encouragements, sincere advices and valuable suggestions made this work interesting and learning for me.

Thanks are also due to my thesis committee members Dr. Mostafa Abd-El-Barr and Dr. Mohamed Jamoussi for their interest, cooperation, advice and constructive criticism.

I also wish to thank the Chairman of COE Department Dr Khalid Al-Tawil for all the support he provided in order to achieve this work. Also, my thanks go to the faculty, and the staff members of the Computer Engineering Department for their encouragements.

Special acknowledgment is due to all my friends at KFUPM for their moral support, encouragements, and good wishes.

I dedicated this work to

my beloved mother

whose prayers, sacrifice, inspiration and love

led to this accomplishment

and to

my loving father

Contents

List of Figures

List of Tables

THESIS ABSTRACT

Name: HAKIM SALAH ADICHE
Title: FUZZY GENETIC ALGORITHM FOR VLSI FLOORPLAN
Degree: MASTER OF SCIENCE
Major Field: COMPUTER ENGINEERING
Date of Degree: November 1997

Floorplanning is an essential step when a building block VLSI design methodology is used. Floorplanning is executed to seek answers to several difficult design problems such as sizes and shapes of blocks, location of pins and pads, etc., while attempting to minimize several objectives. Examples of objectives are minimization of overall area of floorplan, minimization of delays and minimization of total wire length. Floorplanning falls into the class of hard combinatorial optimization problems with multiple objectives and constraints. Iterative algorithms have been found effective search heuristics for such class of problems. However, there has not been agreement on how best one can evaluate the cost of individual solutions in the case of multi-objective optimization. Fuzzy logic is a natural vehicle for solving this problem. Fuzzy logic allows the mapping of different objectives into the interval [0, 1] through a carefully designed set of fuzzy logic values and fuzzy logic rules. This fuzzy mapping transforms the optimization of a vector-valued function into the optimization of a scalar function, resulting from the activation of a number of fuzzy logic rules. In this work, fuzzy Genetic, simulated annealing, and tabu search algorithms (FGA, FSA, and FTS respectively) are implemented. The evaluation of the cost of an individual solution is the result of the application of fuzzy logic rules which combine fuzzy linguistic values defined on the problem domain. The FGA, FSA, and FTS were tested on the floorplanning problem with the same fuzzy cost functions.

King Fahd University of Petroleum and Minerals, Dhahran.

November 1997

خلاصة الرسالة

الآسم : حكيم صالح عديش

عنوان الرسالة : خوارزمية جينية غيرواضحة لتصميم التخطيط السطحي على شكل شرائع المتعلق بالدوائر المتكاملة ذات النطاق الواسع جدا

التخصص : هندسة الحاسب الآلي

تاريخ الشهادة : نوفمبر ١٩٩٧م

التخطيط السطحي من الخطوات الهامة في تصميم الدوائر المتكاملة ذات النطاق الواسع جدا. وهدف التخطيط السطحي هو البحث عن حلول مقبولة لمسألة صعبة ذات تصاميم مختلفة متعلقة بتصغير المساحة السطحية و شكل الوحدات او الخلايا و في نفس الوقت تصغير طول التوصيلات بين الوحدات و تحسين سرعتها.

و يعتبر المنطق الغير الواضح (Fuzzy Logic) وسيلة طبيعية لحل هذا النوع من المسائل الصعبة من خلال تصميم قواعد و قيم غير واضحة مرتبطة بمسألة معينة.

في هذه الرسالة نقوم بتنفيذ خوارزمية جينية غير واضحة. و تقوم التجارب بأستعمال الخوارزمية الجينية الغير الواضحة لحل مسألة التخطيط السطحي على شكل شرائح ثم مقارنة النتائج مع تلك التي نحصل عليها بأستعمال (Simulated Annealing) و (Tabu Search) الغير الواضحين.

درجة الماجستير في العلوم

جامعة الملك فهد للبترول و المعادن

الظهران–المملكة العربية السعودية

نوفمبر ١٩٩٧م

Chapter 1

Introduction

The design of modern electronic circuits is unthinkable without the use of sophisticated design automation tools. These tools usually address a particular design step, formulated as combinatorial optimization problems which are NP-Hard and have several conflicts and noisy objective functions. Iterative algorithms have been found to be very effective to cop with such problems. However, there are no clear criteria that can be used to unambiguously evaluate the fitness of competing individuals. Fuzzy logic provides an effective and easy way to deal with such difficult problems. Two different forms of knowledge exist [18]:

1. Objective knowledge: Frequently used in engineering formulation (e.g. mathematical models).

2. Subjective knowledge: Used to represent linguistic information usually impossible to quantify (e.g. rules and expert information).

Subjective knowledge is always ignored at the front end of engineering designs despite its frequent use to evaluate such designs. The two forms of knowledge can be logically coordinated using fuzzy logic. The first fuzzy logic publication, considered as the seminal paper in this field, was written by Lotfi Zadeh [48], credited as the founding father of the entire theory. The motivation for the use of fuzzy logic can be summarized by the principle of incompatibility, as stated by Zadeh [13]: *"The closer one looks at a real world problem, the fuzzier becomes its solution. Stated informally, the essence of this principle is that as the complexity of a system increases, our ability to make precise and yet significant statement about its behavior diminishes until a threshold is reached beyond which precision and significance become almost mutually exclusive characteristics"*. Since the introduction of fuzzy sets theory in 1965, there has been an explosion of interest both in the mathematical aspects of the theory and its practical impacts. Fuzzy sets theory has been using a good deal of classical mathematical notion [1], and Zadeh's premises run as follows [8]:

"In human thinking, key elements are not numbers nor discrete values, but labels of fuzzy sets, i.e., classes of objects, in which the transition from membership to non-membership is gradual and not abrupt.".

Actually, fuzzy logic foundation has become firmer; its applications have grown in number and variety, and its influence within basic sciences has become more visible and more substantive. Fuzzy logic plays a pivotal role in computing with words [50]. The computation with words finds its motivations when the available

information is too imprecise to state by numbers, and also when there is a tolerance for imprecision which can be exploited to achieve tractability, robustness, and better formulation of reality.

1.1 VLSI Design

The application of fuzzy logic to VLSI design aspect is stated in this work. Indeed, the use of fuzzy set theory is suitable when dealing with problems and situations by their nature ambiguous and not precisely defined.

In an optimization problem with multiple objective functions, it is impossible to optimize every objective. Thus, a trade-off may be of concern which leads to ambiguity due to the non clear relationship between the objectives and the final cost. Fuzzification is suitable for this type of situation. In fuzzy logic approach, optimization of a vector-valued function is replaced by the optimization of a scalar function, which is constructed from satisfaction levels of decision-makers by components of a vector function values [14].

The design of a VLSI circuit is long and complex [24], requiring numerous trade-offs (e.g. area, wire-length, delay). Generally, the concerns of a designer are performance (speed, power, ...) and constraints (area, time, testability, ...). These criteria are closely interrelated and selecting the best design alternatives is not easy.

Fuzzy set theory has been applied in many areas of science and engineering. However, there are very few fuzzy logic applications in VLSI Computer-Aided Design (CAD). This fact is derived from the nature of algorithms used for solving design problems. Most of the models are NP-Hard requiring use of heuristics which are based on human knowledge, acquired through experience. The natural language, which is the basis of fuzzy logic, is a more convenient vehicle for expressing such knowledge. One more reason to consider fuzzy logic approach is the treatment of uncertainties in design data. Fuzzy logic provides more convenient framework to represent such knowledge.

In this work, our purpose is to fuzzify the Genetic Algorithms (FGA) using the floorplanning problem as a testbed. Floorplanning is a generalization of the module placement problem. It consists of determining the orientations as well as block sizes representing general components in order to build up an integrated circuit satisfying the specified constraints such as minimum area, minimum wirelength and minimum delay.

GA is used to search large spaces for a better set of solutions with respect to a certain number of objectives requiring only cost function values to guide the search. Furthermore, it takes a wide global view of the search space than many other methods encountered in engineering optimization [11]. The different methods already used to set up a cost function do not explain the interrelationship between each objective and the final cost. We intend to use fuzzy logic as a tool in order to produce a cost function based on a clear knowlege about how the objectives

to be dealt with are related to the cost. Subjective knowledge in the problem domain is expressed in the form of fuzzy rules. Using fuzzy algebra, these fuzzy rules are translated into objective functions and the computation of the cost value is performed with an objective function extracted from a fuzzy rule. This cost value will serve to guide the search for a good solution whose objectives meet the specified constraints. A number of experiments involving FGA are to be performed. The outcomes will be compared with those obtained by other stochastic iterative algorithms, namely Fuzzy Simulated Annealing (FSA) and Fuzzy Tabu Search (FTS) heuristics.

1.2 Organization of This Thesis

Chapter 1 consists of the introduction to the thesis. It is followed by a review of the existing literature on the floorplanning problem in Chapter 2. Chapter 3 describes the floorplan and the computation of the objective values. The heuristics used in this work are explained in Chapter 4. This is followed by an introduction to fuzzy logic in Chapter 5. Chapter 6 describes the different mechanisms to be applied in order to derive a cost function using fuzzy logic. The experimental results are illustrated and analysed in Chapter 7. Finally, conclusions are drawn in Chapter 8.

Chapter 2

Literature Review

There is an extensive literature on the area minimization problem. Stockmeyer showed that area minimization for general floorplans is strongly NP-complete [35]. Area minimization algorithms can be divided into two classes, those able to handle only a special class of floorplans and those able to handle general floorplans. For slicing floorplans, Otten and Stockmeyer first proved that the minimization problem of a fixed floorplan can be solved in polynomial time. Stockmeyer presented an algorithm of time complexity $O(nd)$, where n is the number of basic blocks, d is the depth of the slicing tree, and each basic block has two realizations. The time complexity is $O(n^2)$ in the worst case, or $O(n \log n)$ for balanced slicing tree with d=O(log n). The worst case time complexity is $O(n^2)$ when the depth of the slicing tree is $O(n)$. The space complexity is the same as the time complexity. If more than two realizations for each basic block are allowed and evenly distributed among the basic blocks then the Stockmeyer's algorithm running time

will be $O(n^2)$.

Other approaches are implemented to solve the floorplanning problem such as simulated annealing [21] where an algorithm (based on simulated annealing) was developped by Wong and Liu [43] to solve the floorplan design problem. This algorithm uses normalized Polish expression representation and is optimized for area and total interconnection length in the final solution. The experimental results indicate a good performance in many test problems.

In [41], a branch and bound algorithm is implemented. This algorithm determines the dimensions of each basic block of a general (non-slicing) floorplan such that the layout is minimized. It can handle large floorplans starting with several possible physical implementations for each basic block.

In [9], floorplanning is implemented in combination with placement. If the layout includes flexible blocks, the placement result can be further optimized by resizing these blocks.

Distributed genetic algorithm is implemented to solve the floorplan design problem [6]. This method is shown to perform better than the simulated annealing approach, both in terms of the average cost of the solution found and the best-found solution.

The floorplan problem is also solved using an analytical method [36]. This method is developed for a general floorplan. It is based on a mixed integer programming model and application of a standard mathematical software. It provides a proven optimal solution for each subproblem solved in the process of constructing of a

global solution. Various objectives functions: area, wire-length, and timing delay are permitted. In [38], an optimal algorithm for floorplan area optimization problem whose run time is less than that of the Wimer algorithm is described. This algorithm deals with large floorplans. It eliminates a large number of implementations which are not relevant to the optimal solution. For several examples where the Wimer algorithm ran for days without terminating, optimal solutions were produced by this algorithm in few seconds. The time complexity, However, is still exponential in the worst case $O(k^3 \log k)$ where k is the number of realizations for each basic block. The time complexity of this algorithm is improved to $O(k^2 \log k)$ through the algorithm presented in [3]. In [5], an approximate algorithm is proposed to deal with large problems with acceptable CPU time and memory requirements. It consists of a linear time algorithm to determine all the needed primitive superblocks. Equivalent basic blocks are found by using Stockmeyer's algorithm if the primitive superblock has a slicing structure and by using branch-and-bound if not. This combined strategy can reduce dramatically the overall run time of the algorithm to find an optimal realization. Another algorithm for slicing structure whose performance is proven by experimental results using industrial benchmark circuits is presented in [46]. Low percentage of dead area and fast CPU times are obtained. A unified framework for enumeration and optimal sizing on a set of sliceable floorplans is also presented. It is based on an adjacency graph derived from circuit connectivity in order to generate sliceable floorplans. Any algorithm computing the minimum floorplan area may be faced

with the problem of insufficient memory in the case where the floorplan is large. A method was devised in [39] to handle this situation. During the computation, whenever the set of non-redundant implementations of a sub-floorplan exceeds a certain predefined size, a subset of the implementations that can best approximate the original set is retained. Two algorithms to select minimum area for rectangular and L-shaped sub-floorplans are presented. In [23], two area minimization methods for general floorplans are proposed. They are generalizations of the classical algorithms for slicing floorplans of Otten and Stockmeyer. These two methods are proved to be better than the branch-and-bound algorithm of Wimer et al. These methods reduce naturally to a polynomial algorithm for slicing floorplans. They are also quite efficient for approximately slicing floorplans. In [28], authors present an algorithm which is particularly effective for floorplans that are approximately slicing and reduces to a polynomial algorithm for slicing floorplan. In [25], an approach based on Genetic Algorithm is proposed for the floorplan area optimization problem. The main advantages of this approach are:

1. Simple implementation,

2. easy trade-off between CPU time and result accuracy, and

3. limited amount of memory to store partial results.

This method, however, produces near optimal results with CPU time requirement comparable with the ones of other iterative approaches. In [4], other techniques are presented. They are used for reducing the time space costs of solving various

floorplan area minimization problems. The reduction is by more than a factor of n and n^2 for rectangular and L-shaped sub-floorplans respectively, (n is the number of modules). In [40], a recent approach is proposed for the problem of floorplan area optimization. This approach is based on the analogy between a floorplan and a resistive network. It consists of a class of zero wasted area that can be achieved under the shape constraint of continuous aspect ratio. Necessary conditions are defined for the realization of zero wasted area floorplan under the shape constraints. If no zero wasted area is achievable, a set of methods are then used to minimize the wasted area. A recent algorithm for area optimization of slicing floorplans is presented in [33]. It is characterized by a time complexity of $O(n \log n)$ and space complexity of $O(n \log n)$, or $O(n)$ where n is the total number of realizations for the basic blocks. It is an improvement of the Stockmeyer's $O(n^2)$ time and $O(n^2)$ space algorithm. This algorithm uses a new data structure for storing and updating realizations.

Chapter 3

Floorplanning

3.1 Introduction

Floorplanning is a process that comes at the early stages of VLSI physical design. The geometry of the basic blocks in terms of width w and height h is roughly estimated but the area of each block is available. In fact, a floorplan consists of an enveloping rectangle partitioned into non-overlapping basic rectangles or modules. For every basic module a set of implementations is given. The goal of the floorplan design optimization problem is to decide topological proximity of the various modules and to find out an implementation for each basic module such that some objectives are met. Figure 3.1(a) is a floorplan with 6 blocks. In this chapter, we give a brief description about floorplanning and the computation of area, wire-length, and delay related to a floorplan.

3.2 Formal Description

Floorplanning is formally described as follows [27]:

1. Let S be a set of n rectangular modules: S=$\{1, 2, ..., n\}$.

2. Let $S1$ and $S2$ be a partition of S where $S1$ and $S2$ represent the set of modules with fixed and free orientations respectively.

3. Let $C_{n \times n}$=$[C_{ij}]$, $1 \leq i, j \leq n$ be an interconnection matrix where C_{ij} is the connectivity between modules i and j.

4. Let have a list of n triplets $(A_1, r_1, s_1)...(A_i, r_i, s_i)$, where A_i is the area of block i. r_i and s_i are lower and upper bound constraints on the shape of block i.

5. Let p and q $(p \leq q)$ be the lower and upper bound on the shape of the rectangle enveloping the n blocks. Then the solution to the floorplan problem would be a set of n non-overlapping rectangles labeled 1, 2, ..., n enveloped by a rectangle R.

A floorplan is feasible if it satisfies the following constraints:

1. $w_i \times h_i = A_i$, $(1 \leq i \leq n)$ (w_i and h_i are the width and height of block i).

2. $r_i \leq \frac{h_i}{w_i} \leq s_i$ for all modules i with fixed orientation ($i \in S1$).

3. $r_i \leq \frac{h_i}{w_i} \leq s_i$ or $\frac{1}{s_i} \leq \frac{h_i}{w_i} \leq \frac{1}{r_i}$ for all modules i with free orientation ($i \in S2$).

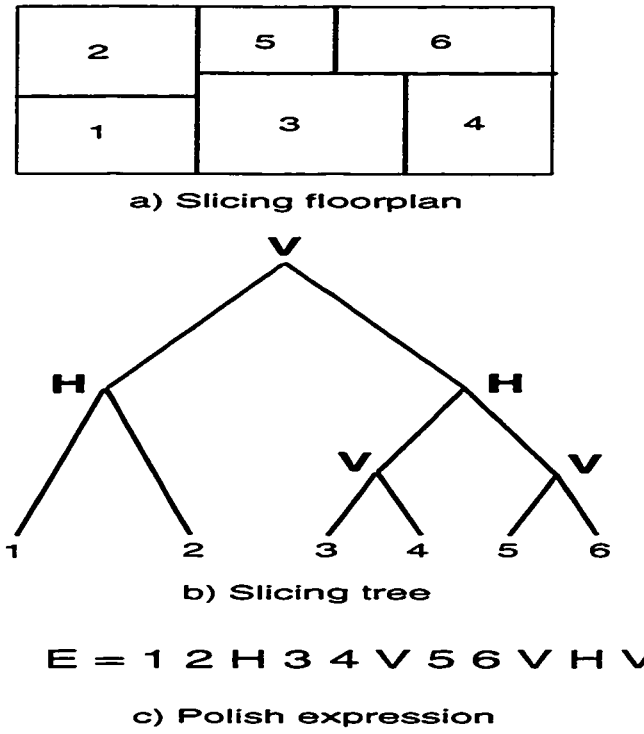


Figure 3.1: (a) Floorplan; (b) its tree representation; (c) its Polish representation.

4. $x_i \geq w_i$ and $y_i \geq h_i$. $1 \leq i \leq n$ where x_i and y_i are the dimensions of basic rectangle i.

5. $p \leq \frac{H}{W} \leq q$ where H and W are the height and width of the enveloping rectangle R.

3.3 Floorplan Representation

The relative positions of the basic modules can be specified by a floorplan tree. The leaves are the basic rectangles, the root is the enveloping rectangle, and the internal nodes are the composite rectangles. Each of the composite rectangles is divided into k parts in a hierarchical floorplan of order k. When k=2, the floorplan

is called a slicing floorplan and the corresponding tree is a slicing tree as shown in Figure 3.1. If a rectangle has to be placed in one of its two possible orientations, then with b rectangles there are 2^b possible configurations of the layout. With the slicing structure, a minimum realization of n rectangles can be determined in $O(n \log n)$ time [33]. A slicing floorplan with n rectangles can be concisely represented by a polish expression with n operands and $n-1$ operators. The operators are either a horizontal slicing line (H) or a vertical slicing line (V). The polish expression is obtained by performing a postorder traversal of the slicing tree. Restricting floorplans to be slicing usually results in more dead space [27], however, this dead space can be removed later on by running a compaction step on the final slicing floorplan [28]. A skewed slicing tree is one in which no node and its right son are the same as illustrated in Figure3.2. A Polish expression is said to be normalized if and only if it has no consecutive H's or V's. The purpose of this classification (i.e., normalized versus non-normalized Polish expression) is to remove the redundant solutions from the solution space in case several Polish expressions correspond to a same slicing floorplan as shown in Figure 3.3 [27].

If the objective function is the floorplan area, then an optimal realization of a floorplan is one that has minimum area. Let R be the set of all realizations of the floorplan. Let A and B be two realizations in R. Let w_A, h_A, w_B, and h_B be the width of A, height of A, width of B, and height of B respectively. B is dominated by A if and only if either $w_A \leq w_B$ and $h_A < h_B$ or $w_A < w_B$ and $h_A \leq h_B$. Let R' be the subset of R obtained by deleting from R all realizations that are dominated

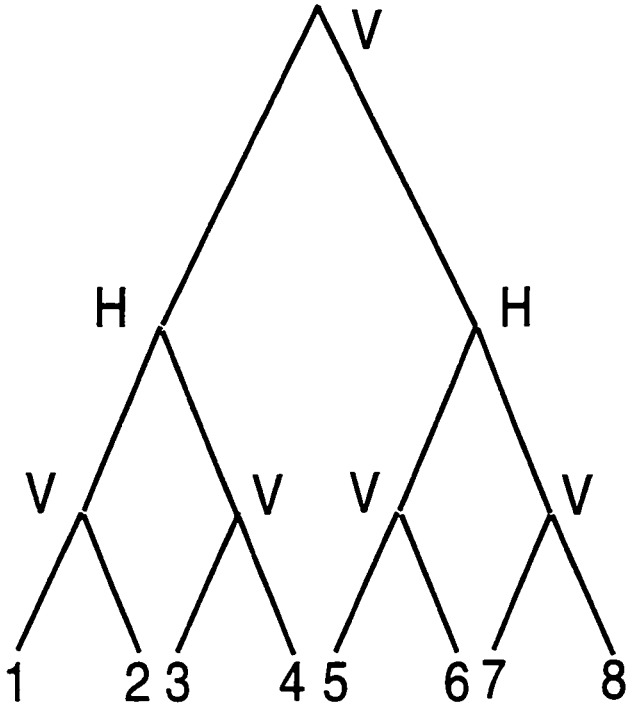
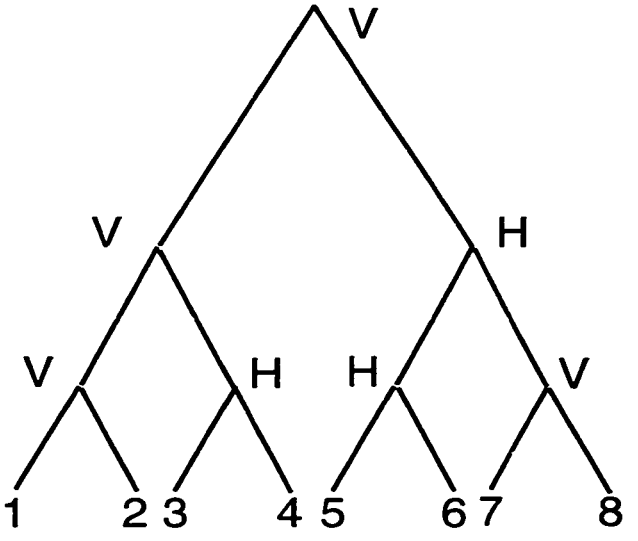


Figure 3.2: Skewed slicing tree.



1 2 V 3 4 H V 5 6 H 7 8 V H V

Figure 3.3: Normalized Polish Expression.

by at least one other realization in R. Two realizations A and B in R' are said to be equivalent if and only if $w_A = w_B$ and $h_A = h_B$. The dominating set of the floorplan is obtained from R' by deleting all except one realization from each class of equivalent realizations.

3.4 Cost function

The floorplanning problem is an NP-hard problem since it is a generalization of the placement problem which is NP-hard. Usually, heuristic techniques are used to identify a good feasible floorplan with desired aspects, not necessarily the best floorplan. The quality of a floorplan can be evaluated on the basis of several objectives such as area minimization, wire-length minimization, routability maximization, delay minimization, power minimization, and combination of two or more of the precedent criteria.

3.5 Objectives values computation

3.5.1 Area

The goal of area optimization is to find the smallest area of the floorplan enveloping rectangle. This process is achieved by combining all the possible realizations of each basic block and determining that combination which leads to the optimum area. Each basic block can have more than two possible realizations. The floorplan

area computation for slicing structure is defined as follows [27]:

Definition 1 Let Γ be a continous curve on the plane. Γ is called a bounding curve if it satisfies the following conditions:

1. it is increasing, i.e., for any two points (x_1, y_1) and (x_2, y_2) on Γ, if $x_1 \leq x_2$ then $y_2 \leq y_1$;
2. Γ lies completely in the first quadrant, i.e., for all $(x, y) \in \Gamma$, $x > 0$ and $y > 0$; and
3. it partitions the first quadrant into two connected regions. The connected region containing all the points (x, x) for very large x is called the bounded area with respect to the bounding curve Γ.

Definition 2 Let Γ and Λ be two bounding curves. Two arithmetic operations on bounding curves are defined:

1. The bounding curve corresponding to Γ H Λ is obtained by summing the two curves along the y-axis, i.e., Γ H $\Lambda = \{ (u, v+w \mid (u, v) \in \Gamma$ and $(u, w) \in \Lambda) \}$.
2. The bounding curve corresponding to $\Gamma V \Lambda$ is obtained by summing the two curves along the x-axis, i.e, $\Gamma V \Lambda = \{ (u+v, w) \mid (u, w) \in \Gamma$ and $(v, w) \in \Lambda \}$.

The following example illustrates the computation of a slicing floorplan area. The dimensions of the basic blocks of the floorplan of Figure 3.4 are listed in Table 3.1.

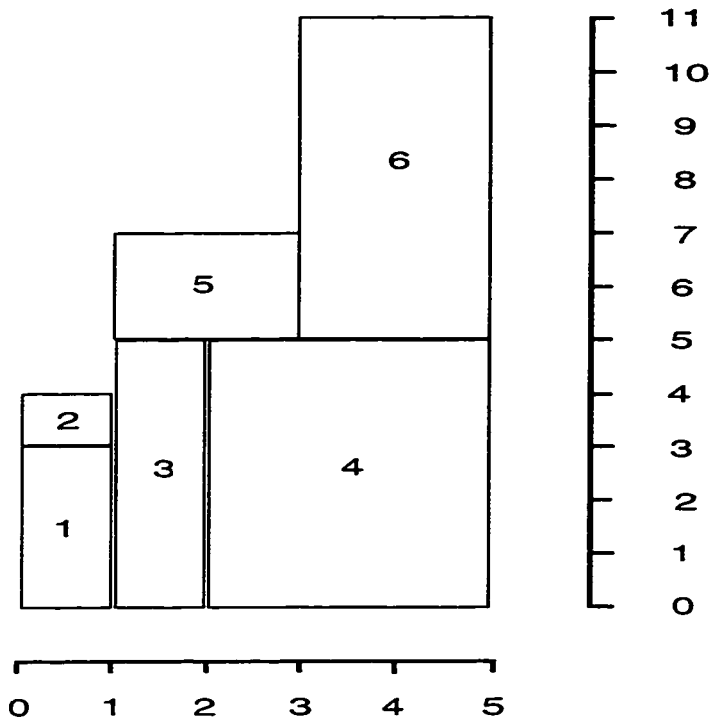


Figure 3.4: Slicing floorplan with initial dimensions for each basic block.

Block	Width	Height
1	1	3
2	1	1
3	1	5
4	3	5
5	2	2
6	2	6

Table 3.1: Initial Dimensions of Basic Blocks.

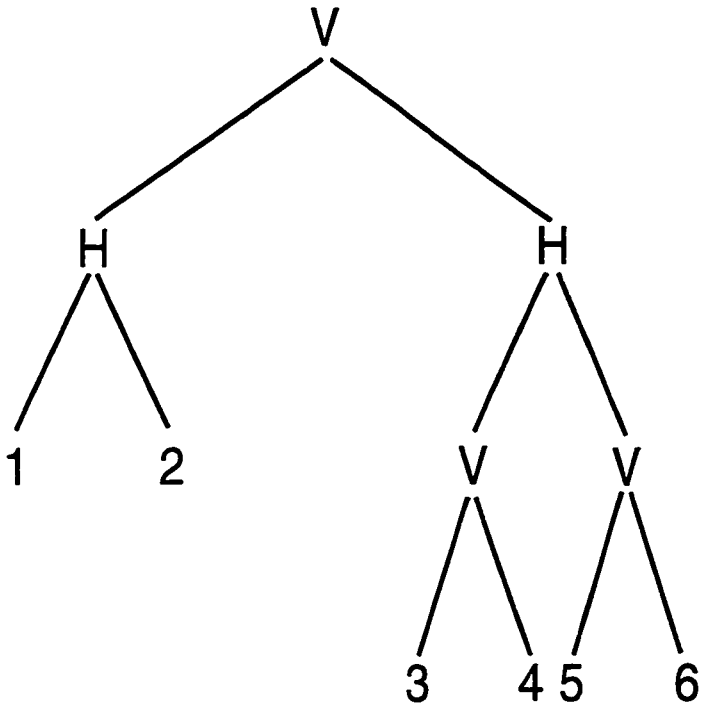


Figure 3.5: Floorplan tree whose Polish expression is E=12H34V56VHV.

All blocks with $w \neq h$ have free orientation as well as flexible shapes. The increment δ with which the width and height vary from w to h and from h to w respectively is determined by the following expression:

$$\delta = \frac{|w-h|}{step}$$

where "step" is a user specified constant that defines the increment which will specify the number of realizations for each basic block. For example, if "step" is set to 2, then there will be three different implementations for each basic block (step + 1 = 3). For the normalized Polish expression E=12H34V56VHV, the floorplan tree is shown in Figure 3.5. A bottom-up traversal to the root using definition 2, as described in Figure3.6, will result in Γ_p={(5, 10), (7, 9.5), (8, 6.7),

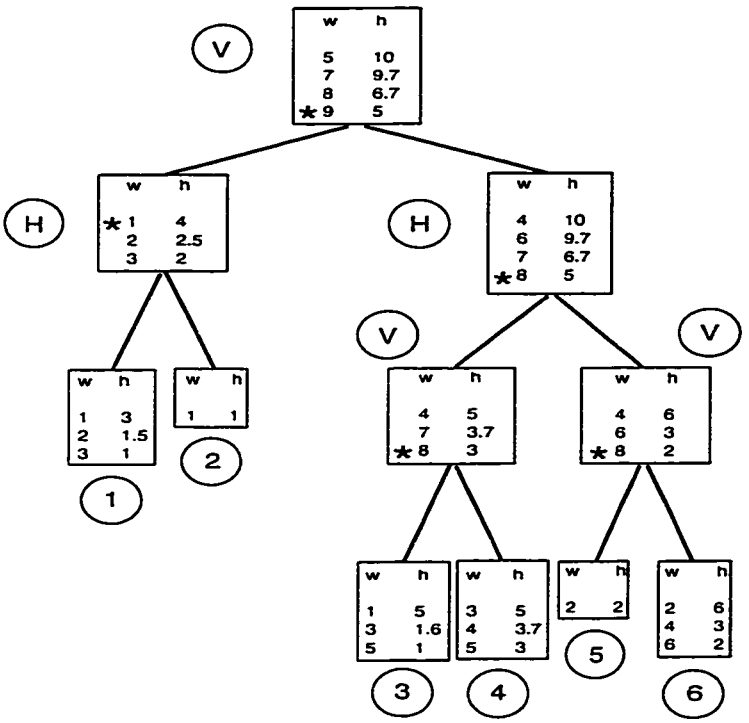


Figure 3.6: Bottom-up traversal from the leaves to the tree.

(9, 5)}. Since 9 × 5 is less then the products obtained with the other possible implementations, then a minimum area enveloping rectangle corresponding to the above Polish expression is a 9 × 5 rectangle. A top-down traversal of the tree from the root to the leaves will specify the dimensions of each basic block such that the area of the resulting floorplan corresponds to the one with dimensions (9, 5). During the bottom-up traversal process, the number of possible realizations for each composite block may become very large. Hence, redundant implementations should be identified and eliminated.

Definition 3 Let (x_1, y_1) and (x_2, y_2) be two possible implementations of a given rectangle. (x_2, y_2) is a redundant implementation of (x_1, y_1) if and only if $x_2 \geq x_1$ and $y_2 > y_1$, or $x_2 > x_1$ and $y_2 \geq y_1$.

Block	Width	Height
1	1	3
2	1	1
3	3	1.6
4	5	3
5	2	2
6	6	2

Table 3.2: Final Dimensions of Basic Blocks.

As an example, the possible dimensions of the composite rectangle 34V are {(4, 5); (5, 5);(6, 5);(6, 5);(7, 3.7);(8, 3);(8, 5);(9, 3.7);(10, 3)}. According to definition 3, the following points are redundant and hence should be eliminated: {(5, 5);(6, 5); (6, 5);(8, 5);(9, 3.7);(10, 3)}. The list of possible realizations becomes {(4, 5); (7, 3.7);(8, 3)}. The basic blocks will have the dimensions as indicated in Table 3.2. The slicing structure corresponding to this solution is shown in Figure 3.7.

Aspect Ratio Constraint

The computation of the minimum area of the enveloping rectangle is not sufficient. The dimensions of the floorplan enveloping rectangle should satisfy a well specified aspect ratio defined within a range bounded by p and q such that the constraint $p \leq \frac{H}{W} \leq q$ is verified, where H and W are the height and width of the enveloping rectangle. It might happen that the dimensions of the floorplan rectangle corresponding to the minimum area in terms of width and height do not satisfy the aspect ratio constraint. If $\frac{H}{W} < p$, then H is incremented by a certain amount ΔH

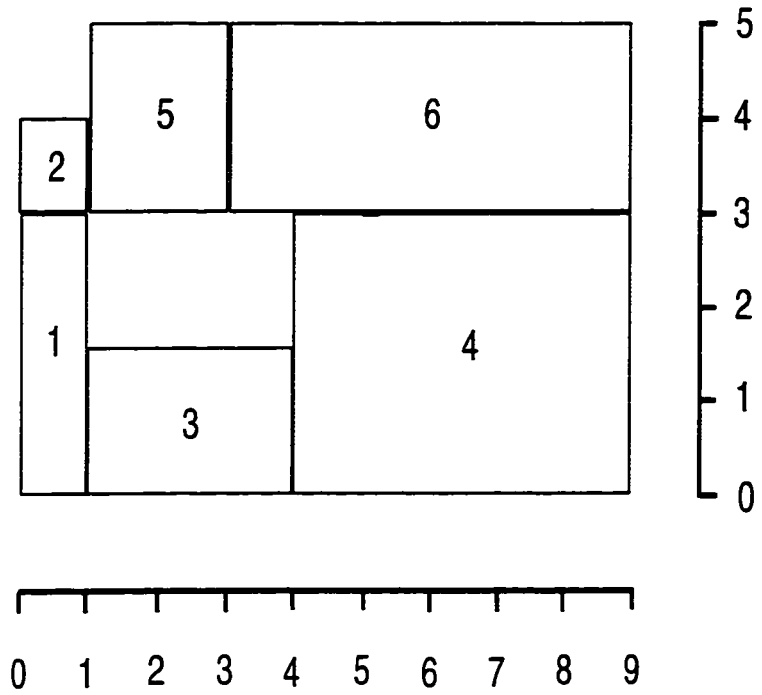


Figure 3.7: Slicing floorplan with final dimensions for each basic block.

such that:

$$\frac{H + \Delta H}{W} \geq p \tag{3.1}$$

where,

$$\Delta H = W \times p - H \tag{3.2}$$

If $\frac{H}{W} > q$ then W is incremented by a certain amount ΔW such that:

$$\frac{H}{W + \Delta W} \leq q \tag{3.3}$$

where,

$$\Delta W = \frac{H}{q} - W \tag{3.4}$$

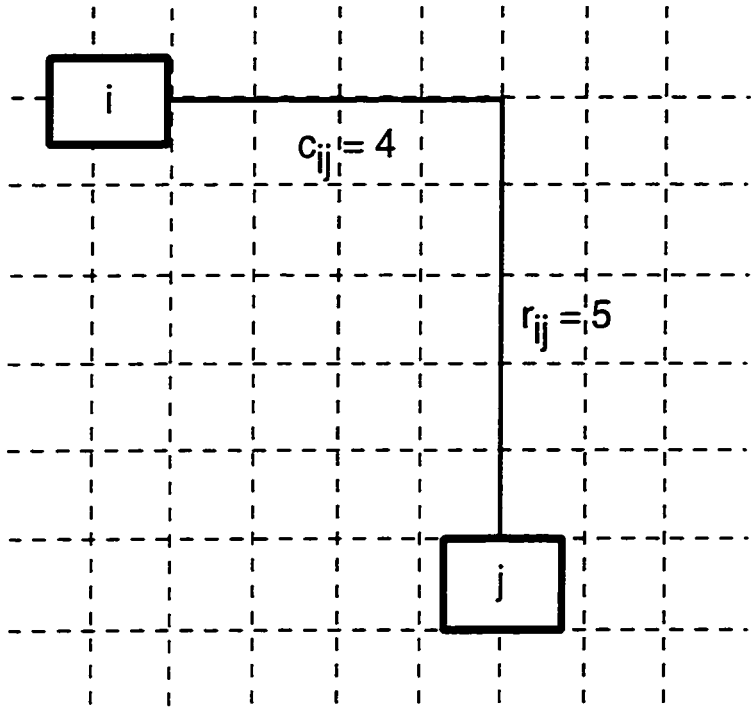


Figure 3.8: Manhattan length estimation.

This process is repeated for all the (W, H) pairs of the enveloping rectangle. The pair giving the minimum area is then selected.

3.5.2 Wirelength

After the area computation, the next step deals with the length estimation of the connections between the different basic blocks. For a two-pin net connecting module i to module j, the Manhattan length of this net is $r_{ij}+c_{ij}$, where r_{ij} and c_{ij} are the number of rows and columns separating the locations of the two modules as shown in Figure 3.8. Since not all nets are two-pin nets, then other methods have to be used. These methods should estimate the length of a multipoint net. In this effect, various techniques are available [27]:

1. Semi-perimeter method.

2. Complete graph.

3. Minimum chain.

4. Source to sink connection.

5. Steiner tree approximation.

6. Minimum spanning tree.

A Steiner tree is the shortest route for connecting a set of pins. A wire can branch from any point along its length to connect other pins of the net. Finding the minimum Steiner tree is proven to be an NP-Hard problem. Several algorithms have been proposed to solve this problem. In [2], a Rectilinear Steiner Tree (RST) algorithm was presented with a time complexity of $O(n^2 \log n)$.

In this work, we estimate the length of connections using the following Steiner approximation technique. The steps are as shown in Figure 3.9. The coordinates x and y of the center of mass are computed using Equations 3.5 and 3.6 respectively:

$$x_c = \frac{x_1 + x_2 + ... + x_n}{n} \tag{3.5}$$

$$y_c = \frac{y_1 + y_2 + ... + y_n}{n} \tag{3.6}$$

where x_1, x_2, ..., x_n are the locations of pins 1, 2, ... n along the x-axis and y_1, y_2, ..., y_n are the locations of the same pins along the y-axis as shown in Figure 3.10.

Procedure (Steiner_Approximation)
    Determine all the pins of a net i
    Enclose all the pins of net i in a bounding rectangle
    Compute the width and the height of the bounding rectangle
    Find the center of mass of this bounding rectangle
    If the width is greater than the height then draw a horizontal
        line passing through the center of mass, otherwise, draw
        a vertical line passing through the same center of mass
    Finally, project all pins of net i on the drawn line.
End_Procedure

Figure 3.9: Steiner approximation procedure.

The Steiner approximation technique used in this work is based on the assumption that each pin is located at the center of its corresponding basic block, for more simplicity in the computation process.

3.5.3 Delay Model

In this work, a "linear delay model" is adopted. This model considers the switching delay and the interconnect delay. The overall delay for any path Π is given by Equation 3.7:

$$T(\Pi) = \Sigma_{c \in \pi} S_c + \Sigma_{n \in \pi} I_n \tag{3.7}$$

where S_c is the switching delay of cell c and I_n is the interconnect delay of net n. The switching delay of a cell c is given by Equation 3.8:

$$S_c = BD_c + LD_c \tag{3.8}$$

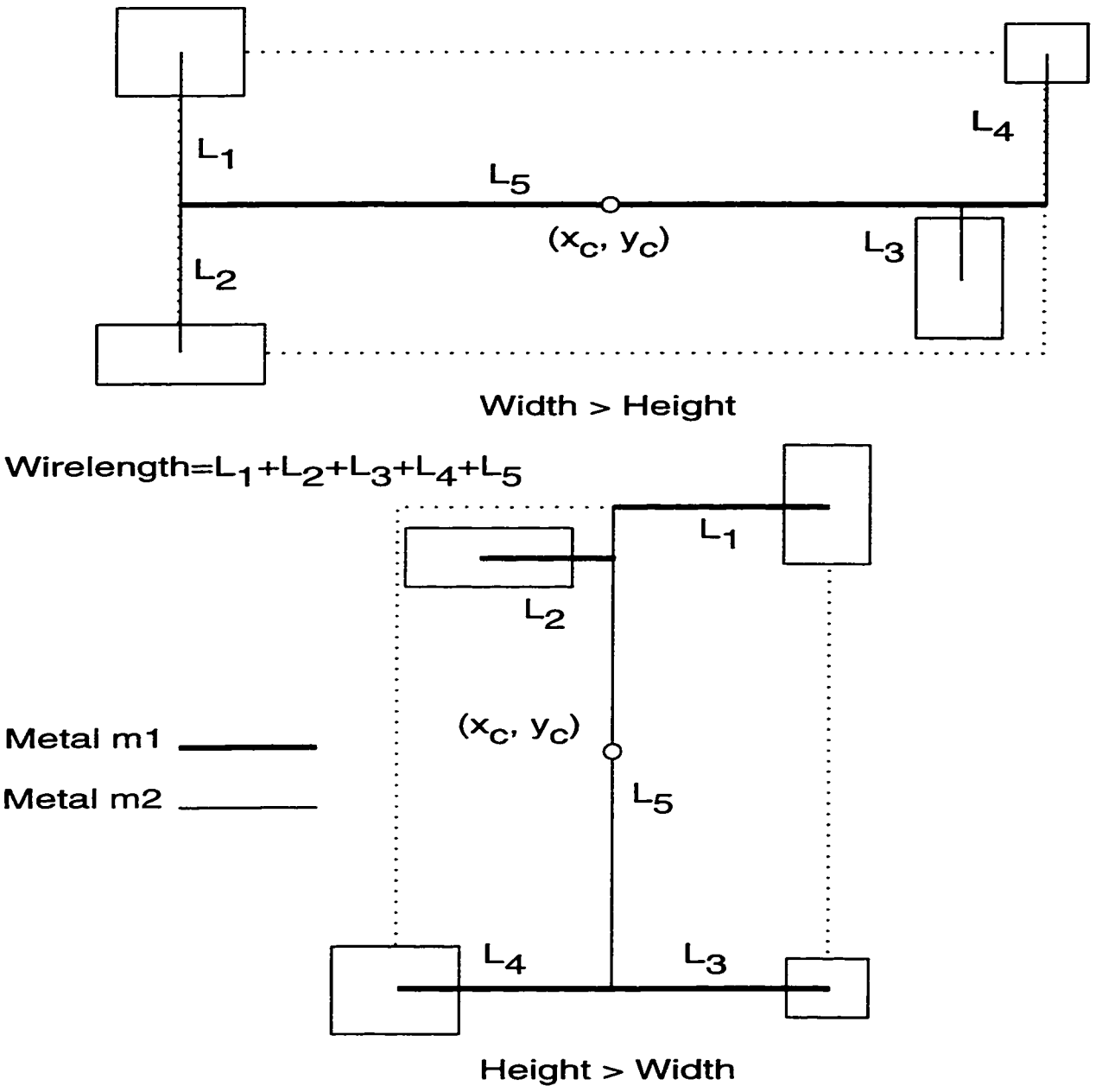


Figure 3.10: Wirelength estimation using Steiner tree approximation.

where BD_c and LD_c are the base delay and the load factor of cell c in nanoseconds (ns).

The interconnect delay of net n is computed using Equation 3.9:

$$I_n = LF_c \times C_n + C_n \times R_n + R_n \times LC_c \tag{3.9}$$

where C_n and R_n represent the capacitance and the resistance of net n respectively. The capacitance C_n is computed as follows:

$$C_n = Area\_Capacitance + Fringe\_Capacitance \tag{3.10}$$

where,

$$Area\_Capacitance = (C_{am1} \times L_{m1} + C_{am2} \times L_{m2}) \times \omega \tag{3.11}$$

and

$$Fringe\_Capacitance = 2 \times ((\omega + L_{m1}) \times Cf_{m1} + (\omega + L_{m2}) \times Cf_{m2}) \tag{3.12}$$

where,

C_{am1} = Plate capacitance per Area of metal m_1.

C_{fm1} = Fringe capacitance per Length of metal m_1.

C_{am2} = Plate capacitance per Area of metal m_2.

Metal_Type	Area_Capacitance ($10^{-4}pF/\mu^2$)	Fringe_Capacitance ($10^{-4}pF/\mu$)
Metal 1	0.26	0.82
Metal 2	0.15	0.85

Table 3.3: Area and fringe capacitance values.

C_{fm2} = Fringe capacitance per Length of metal m_2.

L_{m1} = Length of metal m_1.

L_{m2} = Length of metal m_2.

ω = Width of interconnects.

However, since $C_n \times R_n + R_n \times LC_c$ is much smaller than $LF_c \times C_n$ [28]. , the interconnect delay is approximated by Equation 3.13:

$$I_n = LF_c \times C_n \tag{3.13}$$

The width of interconnects for the circuit technology used in this work is $\omega = 3\mu$. The values for area and fringe capacitances [20] are shown in Table 3.3. Metal m_1 is used for routing horizontal tracks, whereas metal m_2 is used for vertical tracks.

3.6 Floorplanning Techniques

There are three general classes of floorplanning techniques [27]:

Constructive : A feasible solution starts from a seed module or group of modules; then a module (or group of modules) is selected from the set of other modules and added to the partial floorplan. The operation continues until the floorplan is complete. Examples of this technique are cluster growth, partitioning and slicing, connectivity clustering, mathematical programming, and rectangular dualization [27].

Iterative : A floorplan solution is proposed initially. A series of perturbations (neighborhood moves) is applied to this solution until a feasible floorplan is obtained or no better solution can be expected. Examples of iterative techniques applied to floorplanning are simulated annealing, force directed interchange/relaxation, and genetic algorithm [27].

Knowledge-based : It consists of implementing an expert system with the following parts:

1. knowledge base with all the data describing the floorplan problem,
2. rules to use the data stored in the knowledge base,
3. an inference engine which controls the application of the rules to the knowledge base.

3.7 Conclusion

Through the literature related to the floorplanning problem, we notice that some of the proposed techniques have been merged to improve the solution in many aspects. For example, Stockmeyer's algorithm can be used to obtain, in polynomial time, a minimum area realization when the floorplan has a fixed slicing structure and some or all basic blocks have more than two possible implementations each. A number of approaches have also been applied to solve the floorplanning problem such as analytical method based on a mixed integer programming, simulated annealing, and genetic algorithm (distributed and sequential).

Chapter 4

Iterative Algorithms

4.1 Introduction

Many popular algorithms used in physical design automation are iterative. A value of the objective function is used to compare results of consecutive iterations and to select a solution based on the best value of the objective function. Examples of these iterative algorithms include simulated annealing (SA), genetic algorithms (GA), and tabu search (TS) [30]. In this chapter, we give a brief introduction about each of these iterative algorithms, namely SA, GA, and TS.

4.2 Genetic Algorithms

Genetic algorithms were first proposed by John Holland in 1975. They belong to the class of computational models that mimic natural evolution to solve a wide

variety of problems [15, 26]. They have emerged as practical, robust optimization and search methods. GA search methods are rooted in the mechanisms of evolution and natural genetics. Like evolutionary strategies, genetic algorithms draw inspiration from the natural search and selection processes leading to the survival of the fittest individuals [34]. They have been successfully used to solve some hard VLSI design problems such as cell placement [27] and floorplanning [28].

4.2.1 GA Features

There are several differences between GA and other classical optimization methods [11], as listed below:

1. GAs work with a coding of the parameter set, not the parameters themselves.
2. They search from a population of points, not a single point.
3. They use payoff information, not derivatives or other auxiliary knowledge.
4. They use probability transition rules, not deterministic rules.

4.2.2 GA Terminology

1. A chromosome encodes specification of an organism.
2. An organism is made up of one or more chromosomes.
3. A genotype is a complete set of chromosomes.
4. A gene is the basic element of a chromosome.

5. Alleles are the different values of a gene.

In GA, a chromosome is represented as a string of symbols. If only one chromosome is dealt with, then the chromosome and the genotype are the same.

4.2.3 GA Mechanisms

GA starts from an initial (usually random) set of possible solutions called a *population*. Each individual in the population is called a *Chromosome* and represents a solution to the problem. A population of individuals is allowed to evolve through successive generations. Each iteration of GA generates a new population with usually better overall genetic characteristics. Highly fit individuals usually survive to the next generation. Unfit individuals are replaced by new individuals (offsprings) obtained by inheriting features (genes) from two individuals of the current population via a *crossover* operator or by modifying an individual using a *mutation* operator. The computation of the fitness is based on the use of a suitable cost function applied to each individual in the population. The individuals to be included in the next generation are then probabilistically selected according to the fitness values from the set comprising the current generation and the newly formed individuals. A constant number of individuals are selected so that the maintained population is of fixed size. After a certain number of generations, the process is terminated and the best solution encountered is reported.

1	2	H	3	4	H	V	5	6	V	H
5	6	H	2	4	V	3	H	1	V	V
6	3	V	1	4	V	2	H	V	5	H
2	5	V	3	6	H	H	1	V	4	H
6	4	V	2	H	3	5	V	1	H	H
4	1	V	2	H	3	V	5	H	6	V
3	4	H	1	5	V	2	H	6	V	H
1	2	H	3	4	V	3	H	1	V	V
5	6	H	2	4	H	V	5	6	H	H

Figure 4.1: A Sample population where each row is a chromosome representing a particular slicing floorplan.

1. *Encoding*: Encoding is the mechanism which links the GA with the problem at hand. In the floorplanning case, the encoding consists of a Polish expression with block labels and operators (H and V). The Polish expression shows how the blocks are positioned with respect to each other. A sample population is shown in Figure 4.1.

2. *Reproduction*: Reproduction is a process that determines which individuals are fit to mate.

3. *Crossover*: Crossover operation acts as an accelerator of the search process and combines building blocks of good solutions from diverse chromosomes. The choice of parents for crossover is probabilistic. The larger the fitness of

a certain chromosome, the greater is its chance of being selected as one of the parents for crossover. The *roulette-wheel* method is one way to choose the parents for crossover.

4. *Mutation*: Mutation is the means by which new genes are injected into some of the individuals of the population. It consists of altering in a random way the value of a chromosome position.

5. *Inversion*: Inversion operation consists of selecting randomly two positions over the length of a chromosome and then laterally inverting the string between them.

6. *Selection*: Selection step determines which individuals are fit to survive. A set of new solutions is obtained after applying genetic operators on the solutions of the old population. Within each population, a number of solutions have to be selected according to their fitness values. One way is to pick the M best solutions for the new population, where M is the population size.

4.2.4 Roulette-Wheel

A roulette wheel of M slots sized in proportion to the fitnesses of the individuals. Each time another offspring is required, a simple spin of the weighted roulette wheel yields the reproduction candidate. Higher fit chromosomes have a better chance of producing offspring for the succeeding generation. Individuals with low fitness values have a lower probability to be selected. The roulette-wheel can be

implemented on a computer using the following algorithm steps [27]:

1. Construct a list of the fitness values of all individuals in the population.

2. Create another list with fitness values replaced by the addition of a given value to all its precedent fitness values of elements. in the above list.

3. Generate a uniformly distributed random number between 0 and the total of all the fitnesses in the population.

4. Return the first individuals whose fitness value is equal to or greater than the random number generated.

Other selection schemes have also been investigated such as deterministic sampling, remainder sampling without replacement. stochastic sampling with replacement, etc.[11].

4.2.5 GA Procedure

The Genetic Algorithm is shown in Figure 4.2 [29]: The algorithm starts with an initial population whose size is M. Pairs of individuals from the population are chosen. The choice of an individual for mating is proportional to its fitness value. Individuals with higher fitness values have a greater chance of being selected for mating. N_o new offsprings are generated by applying crossover. Mutation and inversion are also applied with a low probability. Experiments indicate that performing mutation after selection leads to sizeable improvement with respect

```
Algorithm (Genetic_Algorithm)
    M = Population size.                    (*# Of possible solutions at any instance.*)
    N_g= Number of generations.             (*# Of iterations.*)
    N_o= Number of offsprings.              (*To be generated by crossover.*)
    P_μ= Mutation probability.
    P_inv=Inversion probability.
    P ← M.                                  (*Construct initial population P.*)
    for j = 1 to M
            Evaluate fitness of P.
    end for
    for i = 1 to N_g
            for j = 1 to N_o
                    Select parents according to their fitness.
                    Generate offsprings by crossover.
            end for
            for j = 1 to N_o
                    Evaluate fitness of each offsprings.
            end for
            Select best M solutions from parents and offsprings.
            for j = 1 to N_o
                    Apply mutation with probability P_μ except on the best.
                    Apply inversion with probability P_inv except on the best.
            end for
    end for
    Return highest scoring configuration in P.
End
```

Figure 4.2: Structure of the basic genetic algorithm.

to the quality of results [28]. This is due to the fact that solutions with inferior cost values are less likely to be selected for the next generation and also this will increase the likelihood that the search procedure gets trapped into local minima. In contrast, the application of mutation after selection allows the effects of the mutation to be inherited by next generation. The offsprings are evaluated on the basis of fitness, and a new generation is formed by selecting some of the parents and some of the offsprings. The procedure is executed N_g times, where N_g is the number of generations. After a certain number of generations or when no more improvement can be achieved, the most fit individual is returned as the desired solution. In the above algorithm, both the mutation operation and the inversion operation are performed after selection of the best M solutions from the parents and the offsprings with a probability P_μ and P_{inv} respectively. The cycles through which the GA operates are illustrated in Figure 4.3;

4.2.6 Crossover

Depending on the problem under study, the crossover operator may have different schemes. In the case of the floorplanning problem, a number of ways to implement the crossover are proposed in [6].

Crossover Operator 1 (CO1)

In this scheme, the operand from parent $P1$ is copied into the corresponding positions in the offspring O. Then, the operators from parent $P2$ are copied by

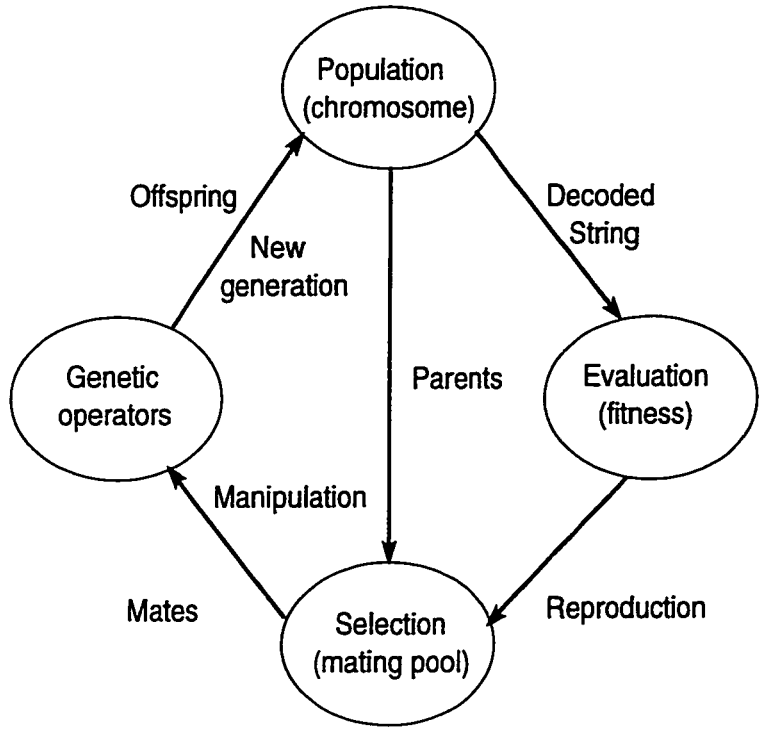


Figure 4.3: The GA cycle.

making a left-to-right scan to complete O as shown in Figure 4.4. Each offspring inherits the block structure of one of the parents.

Crossover Operator 2 (CO2)

CO2 starts out by copying the operators from parent $P1$ into the corresponding positions in offspring O. Then, it completes the construction of O by copying the operands from parent $P2$ by making a left-to-right scan. By propagating the groups of operators from $P1$ into O, CO2 produces an offspring having the same overall slicing structure as that of $P1$ as shown in Figure 4.5. Both CO1 and CO2 crossover operators are designed so that the resulting string is a valid Polish expression for a slicing tree.

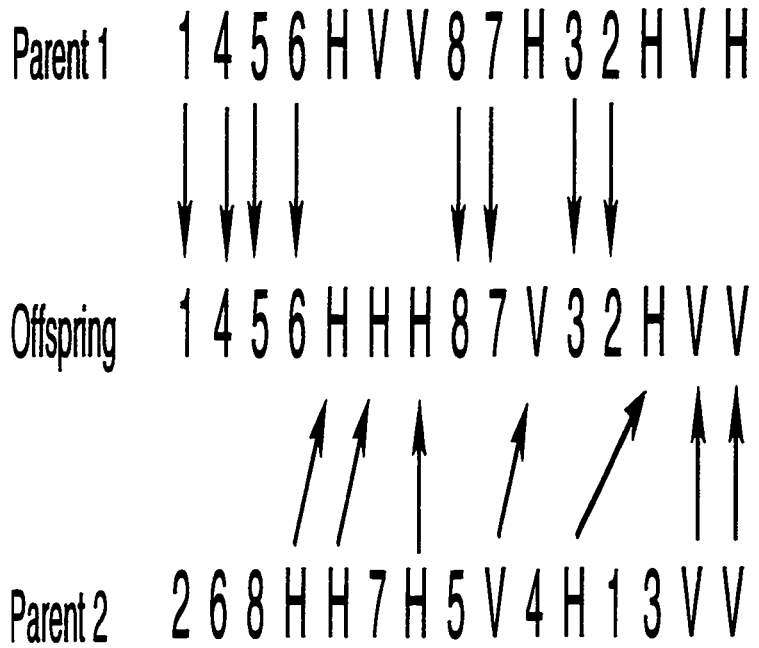


Figure 4.4: Crossover Operator 1.

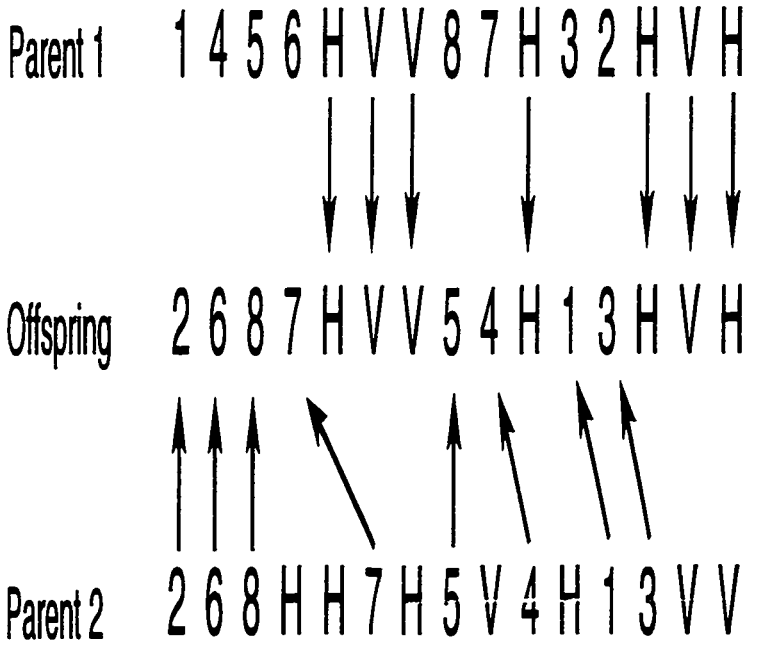


Figure 4.5: Crossover Operator 2.

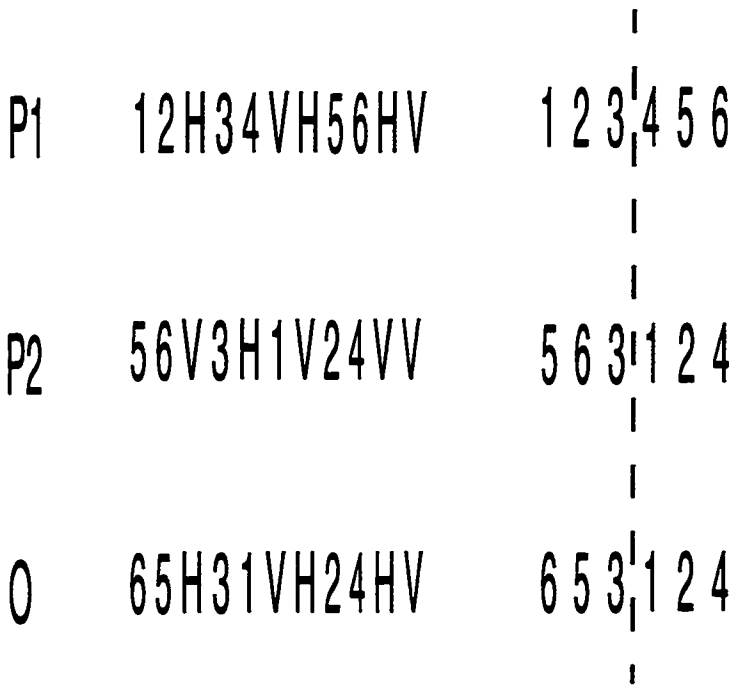


Figure 4.6: Partially Mapped Crossover (PMX).

Partially Mapped Crossover (PMX)

Another scheme for crossover is called Partially Mapped Crossover [32]. As shown in Figure 4.6, a cross-point is randomly selected in the list of operands of parents $P1$ and $P2$. The sub-string to the right of the cross-point in parent $P2$ is copied to the offspring O. Next, the first operand of parent $P1$ is copied to the offspring O provided that this operand is not already present in the offspring, otherwise, a gene from parent $P1$ having the same position as the previous gene in offspring O is considered. This operation continues until the offspring O is fully obtained.

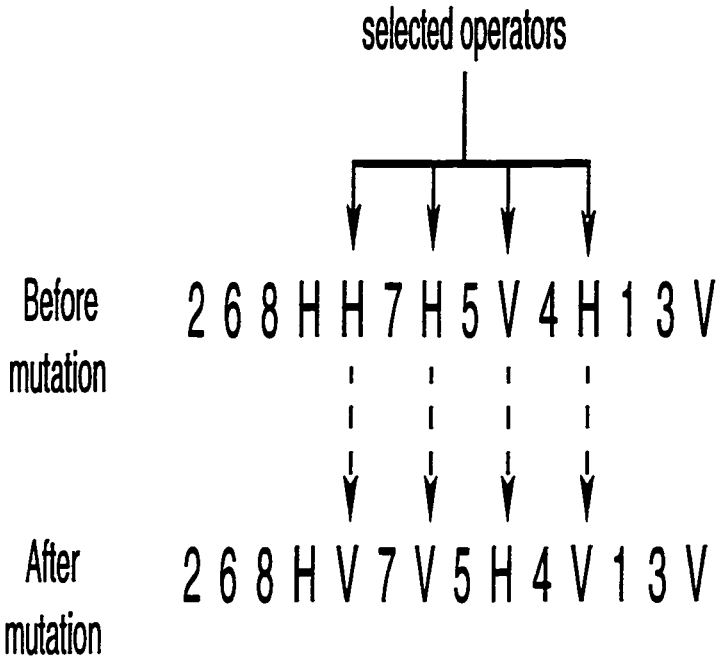


Figure 4.7: Invert a chain of operators.

4.2.7 Mutation

Several schemes for the mutation operation are proposed in the literature. Some of them have been used for the purpose of dealing with the floorplanning problem [42].

Inversion of a Chain of Operators

A series of adjacent operators is selected and then inverted as shown in Figure 4.7.

Swap Adjacent Operand/Operator

In this mutation scheme, two adjacent operator and operand are selected and then swapped as shown in Figure 4.8. The problem with this scheme is that it may

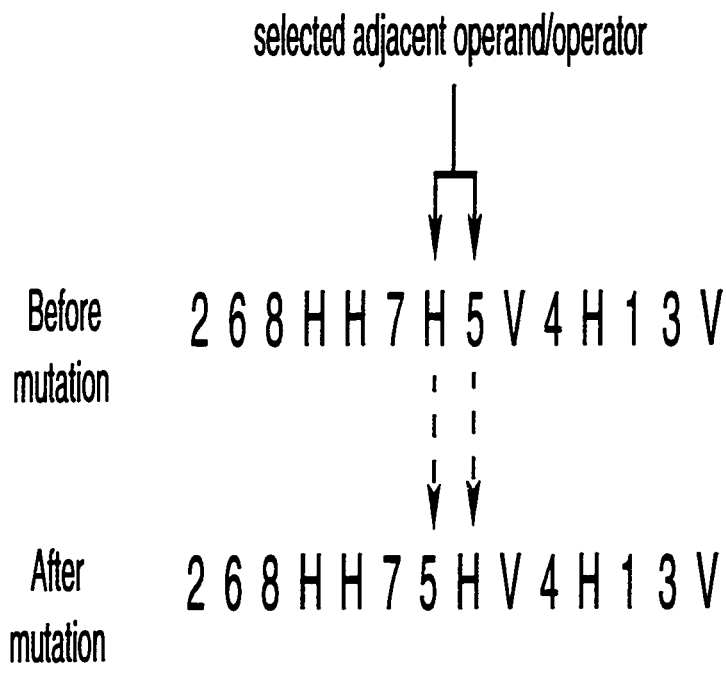


Figure 4.8: Swap adjacent operand/operator.

generate an invalid Polish expression. Hence, appropriate checks are made before performing such an operation.

Swap Two Randomly Selected Operands

This mutation operation is based on swapping two operands selected randomly. As shown in Figure 4.9, the second polish expression is obtained after swapping both operands (3 and 6) chosen randomly in the first one.

4.2.8 Inversion

The proposed scheme for the inversion operation consists of making a full right-to-left or left-to-right rotation of the operands such that the first and the last

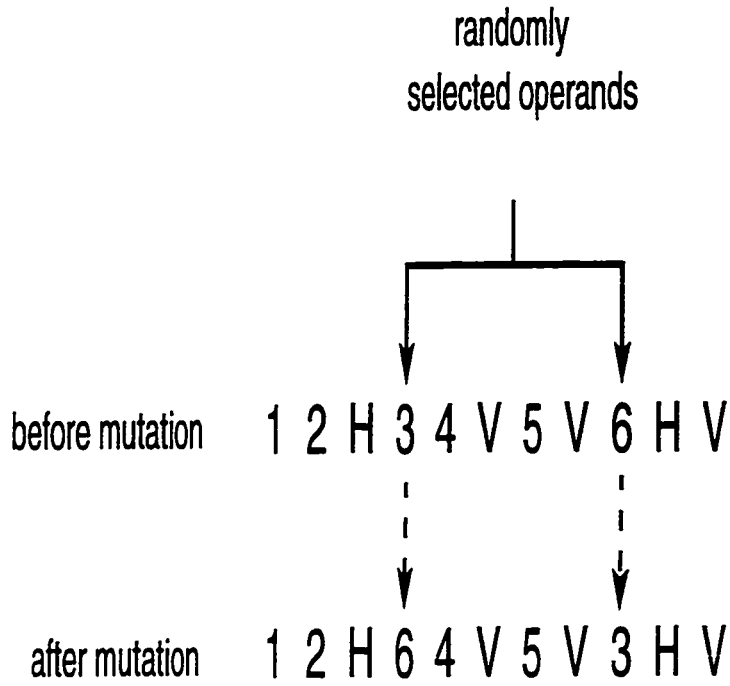


Figure 4.9: Swapping of two randomly selected operands.

operands in the initial string become the last and the first operands respectively in the new string as demonstrated in Figure 4.10.

4.3 Simulated Annealing (SA)

Simulated annealing is an *adaptive* heuristic and belongs to the class of *non-deterministic* (iterative) algorithms. It was first introduced by Kirkpatrick, Gelatt and Vecchi in 1983 [16]. It has been applied to several combinatorial optimization problems from various fields of science and engineering such as VLSI design, image processing, network topology design, etc. Also, the simulated annealing technique has been applied to the floorplan design problem [43]. An iterative improvement

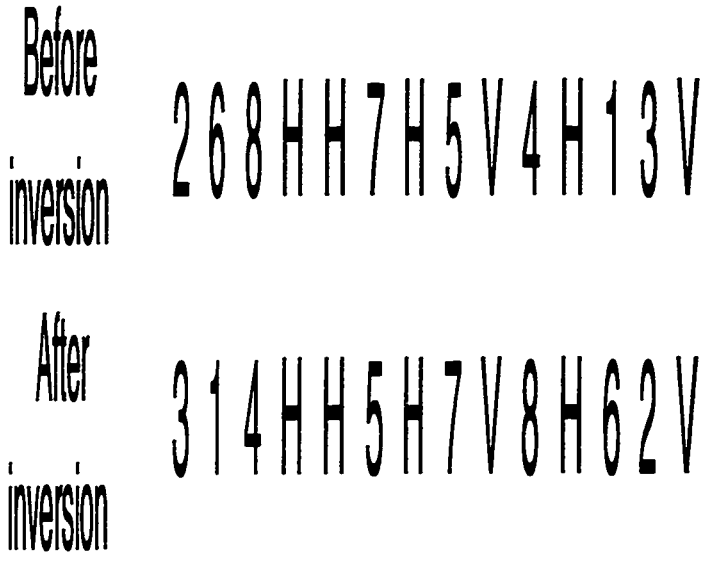


Figure 4.10: Inversion.

scheme starts with some given state, and explores the local neighbourhood of the state for better solutions. It will move to that local neighborhood from the current state if the former has a better cost. If all the local neighbors have inferior costs, the algorithm is said to have converged to a local optimum. Simulated annealing creates a non-zero probability to move a state toward a higher-cost state so it may escape a local optimum [17]. In the initial phase, a higher temperature should be used so that it is easier for the states to escape from a local optimum. The temperature will be gradually decreased according to a pre-specified schedule. After the state is placed near the neighborhood of the global optimum, a lower temperature will more likely make the state move closer to the exact global optimum.

4.3.1 Algorithm

The simulated annealing technique is applied to the floorplan design problem as described by the algorithm shown in Figure 4.11 [30]. The temperature T is decreased using a fixed ratio, i.e. $T_k = r \times T_{k-1}$ when $0<r<1$. In [43], the ratio r was set to 0.85. The algorithm starts with any initial Polish expression which is usually far from the optimal solution. At each temperature, enough moves are tried until there are N down-hill moves or the total number of moves exceeds $2N$. The moves are, in fact, different perturbations applied on the Polish expression with probability P_{pert}. The annealing process terminates if the number of accepted moves is less than 5 % of all moves made at a certain temperature or the temperature is low enough. The value of T_0 is determined from a sequence of random moves as shown in the algorithm of Figure 4.12 [43]. We should have $e^{-\Delta_{avg}/T_o}=P \approx 1$ so that there will be a reasonable probability of acceptance at high temperatures. This suggests that $T = -\frac{\Delta_{avg}}{ln(P)}$ (where ln is the natural logarithm) is a good choice for T_o [43].

4.4 Tabu Search

Tabu search technique is an iterative algorithm used to solve combinatorial optimization problems. Tabu search was first proposed by Fred Glover. It is a *metaheuristic* which can be used to guide the search in complex solution spaces and to direct other heuristic procedures as well [30]. One important characteristic

```
Algorithm (Simulated_annealing)
    (* S is the initial solution. *)
    (* New_S is the intermediate solution. *)
    (* Best is the best solution. *)
    (* At is the number of bad moves at a specific temperature. *)
    (* Mt is the number of all moves at a specific temprature. *)
    (* T_o is the initial temperature. *)
begin
    T=λ_T×T_o    (* λ_T < 1 and T_o is the initial temperature. *)
    Best=S.
    Cost=Cost_Function(area, wirelength, delay, S).
    MaxCost=Cost.
    Repeat
        At=Mt=0.
            Repeat
                P_pert=random()          (* P_pert is the probability of perturbation. *)
                select perturbation with probability P_pert.
                New_S=Perturb(S).
                New_Cost=Cost_Function(area, wirelength, delay, New_S).
                h=Cost - New_Cost.
                Cost=New_Cost.
                If (h < 0) Then
                    S=New_S.
                    If (New_Cost > MaxCost) Then
                        MaxCost=New_Cost.
                        Best=S.
                    End If
                Else
                    If (random() < e^(-h/T)) Then
                        S=New_S.
                        At=At+1.
                    End If
                End If
                Mt=Mt+1.
            until(At > N or Mt > 2N)
            T=r×T* r is the cooling rate and T is the actual temperature. *)
    until(T < 0.0001 or At/Mt < 0.05)
End
```

Figure 4.11: Simulated annealing algorithm.

```
Algorithm (Initial_temperature)
    (* S is the initial solution. *)
    (* New_S is the intermediate solution. *)
    (* Max_Moves is the maximum number of allowed moves. *)
    (* T_avg is the average temperature of bad moves. *)
    (* T_o is the initial temperature. *)
    (* n is the number of bad moves. *)
    (* P ≈ 1. *)
    (* P_pert is the probability of perturbation. *)
begin
    n=0
    down_hill=0.
    Cost=Cost_Function(area, wirelength, delay, S).
    New_S=S.
    For i=0 to Max_Moves
            P_pert=random().
            select perturbation with probability P_pert.
            New_S=Perturbate(S).
            New_Cost=Cost_Function(area, wirelength, delay, New_S).
            h=Cost-New_Cost.
            Cost=New_Cost.
            If (h > 0) Then
                down_hill=down_hill+h.
                n=n+1.
            End If
    End For
    T_avg=down_hill/n.
    T_o=-T_avg/log(P).
End
```

Figure 4.12: Initial temperature algorithm.

of Tabu search is that it can easily escape local optimality. Tabu search is based on a flexible memory approach. Three types of memory are considered with this technique:

1. a short-term memory process which constitutes the core of the TS algorithm,
2. an intermediate-term memory process for intensifying the search in a restricted region of the search space. and
3. a long term memory process whose goal is to globally diversify the search.

4.4.1 Algorithm

A simple implementation of the tabu search algorithm involving a short-memory process is shown in Figure 4.13 [30]. The algorithm starts from an initial solution S (current solution). A number of neighbor solutions V^* are generated by perturbing, each time, the current solution S until a neighborhood of S, N(S), is built. Since we are maximizing, the cost is computed for each neighbor solution. The maximum cost solution, corresponding to S^*, is then retained. If the attribute of the move from S to S^* is not in the tabu list then the move is accepted, the best solution is updated with the tabu list and the aspiration level and the i^{th} iteration number is incremented. In the other case where the attribute of a move is in the tabu list, we have to avoid returning to a recently visited solution for the next k iterations (k being the size of the tabu list). However, this may prevent moves to unvisited solutions. To relax the action of the tabu list in presence of a

```
Algorithm (Tabu Search)
    (* X     : Set of feasible solutions. *)
    (* S     : Current soluton. *)
    (* S*    : Best admissible solution. *)
    (* N(S)  : Neighborhood of S ∈ X. *)
    (* V*    : Sample of neighborhood solutions. *)
    (* T     : Tabu list. *)
    (* AL    : Aspiration Level. *)
begin
    Start with an initial feasible solution S ∈ X.
    Initialize tabu list and aspiration level.
    For fixed number of iterations Do
            Generate neighbor solutions V* ⊂ N(S).
            Find best S* ∈ V*.
            If move S to S* is not in T Then
                    Accept move and update best solution.
                    Update tabu list and aspiration level.
                    Increment the i^th iteration number.
            Else
                    If c(S*) > AL Then
                        Accept move and update best solution.
                        Update tabu list and aspiration level.
                        Increment the i^th iteration number.
                    EndIf
            EndIf
    EndFor
End
```

Figure 4.13: Tabu search algorithm.

sufficiently good move, the aspiration level criterion (AL) is used to override the tabu status of moves to appropriate regions of the space with unvisited solutions. The value of AL is nothing but the best cost (maximum value) computed in the current neighborhood, i.e., $N(S)$ of S.

4.5 Conclusion

Genetic algorithms are characterized by a parallel search of the state space as against a point-by-point search by the conventional optimization techniques. The application of genetic algorithm for any problem requires a representation of the solution to the problem, a choice of genetic operators, an evaluation function, a selection mechanism and determination of probabilities controlling the genetic operators. Simulated annealing and Tabu search share some of the requirements of the genetic algorithm such as a proper encoding of the solution and a cost evaluation function.

Chapter 5

Introduction To Fuzzy Logic

5.1 Introduction

This chapter describes the basics of the fuzzy set theory that we intend to use in order to build our fuzzy iterative stochastic algorithms. Let us consider, for example, a "class of numbers near two". This class is fuzzy because of the imprecise meaning of the word "near". Sets of such nature very often involve adjectives, verbs and adverbs or combination of them which are ill-defined (not sharply defined) in their descriptions. For example, if we say "tall person," we cannot clearly determine who is tall or who is not tall. With a range of height of 140 cm to 200 cm, the degree to which height x can be labelled "tall" is μ, i.e., we make height x correspond to degree μ ($0 \leq \mu \leq 1$). If the horizontal axis is x and the vertical axis is μ, the graph would be drawn as in Figure 5.1. The horizontal axis is the quantification of the word, the expression of height in one-dimensional space, and

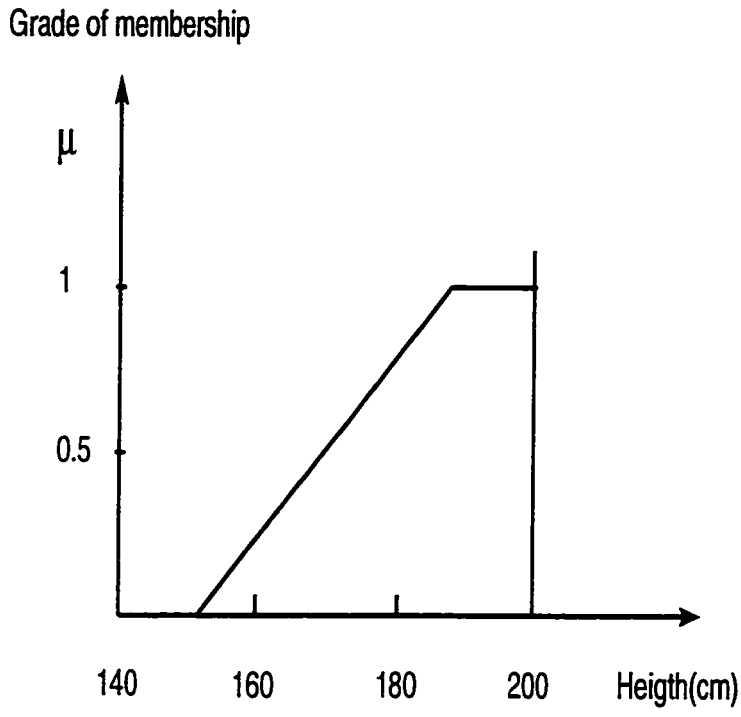


Figure 5.1: Quantifying graph for the adjective "tall"

the vertical axis is the quantification of the degree of ambiguity.

5.2 Fuzzy Sets

More often, the classes of objects encountered in the real world do not have defined criteria of membership. The concept of a *fuzzy set*, that is, a class with a continuum of grades of membership is used to deal with such classes. A fuzzy set F defined on a universe of discourse U is characterized by a *membership function* $\mu_F(x)$ which takes on values in the interval [0, 1]. A fuzzy set is a generalization of an ordinary subset whose membership function only takes on two values, zero or unity. A membership function provides a *measure of the similarity degree* of an

element in U to the fuzzy subset. A fuzzy logic element can reside in more than one set with different degrees of similarity. This kind of word representation is called *quantification of meaning.*

5.3 Description of a Fuzzy System

Fuzzy systems contain *fuzzifiers* which convert inputs into their fuzzy representations, and *defuzzifiers* which convert the output of the fuzzy logic process into *crisp* solution variables [7]. The values of a fuzzified input execute all the rules in the knowledge base that have the fuzzified input as part of their premise. This process generates a new fuzzy set representing each output or solution variable. Defuzification creates a value for the output variable from that new fuzzy set. In Figure 5.2, a typical fuzzy system is represented [7]. The input is read from an external source and fuzzified before being processed by the fuzzy logic. The output of the process logic is then defuzzified. One area where fuzzy set theory has made a considerable contribution is in the problem of Multi-Criteria Decision Function (MCDF) [44].

5.4 Fuzzy Propositions

Fuzzy propositions are propositions including fuzzy predications like "it will probably rain tomorrow" and "x is a small number" [37]. Generally, they are written as follows:

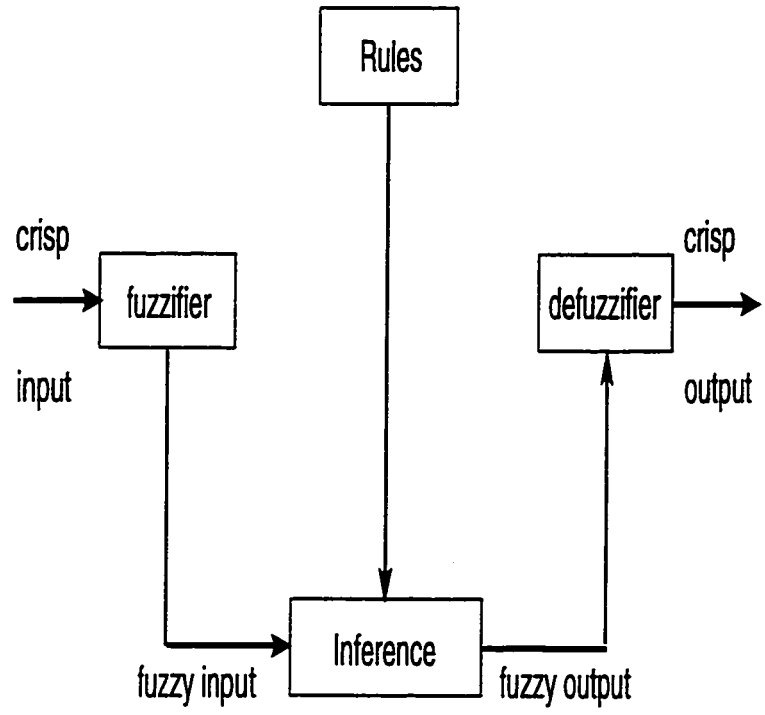


Figure 5.2: Fuzzy system components

$$x \text{ is } A$$

where A is a fuzzy predicate and is called the *fuzzy variable.* Fuzzy variables are also called linguistic variables and are expressed in terms of fuzzy sets. It is possible to modify the predicate of "x is a small number" to the form "x is a very small number." Words like "extremely" and "very," which change the predicate in this way, are called modifiers (hedges) and are indicated by the symbol m.

If "x is A" is modified with m, we use the expression x is m A. The negation of A, "not," can also be thought of as a modifier. The fuzzy sets mA from fuzzy set A are derived as follows:

$$very\ A = A^2 \tag{5.1}$$

$$more\ or\ less\ A = A^{1/2} \tag{5.2}$$

$$not\ A = 1 - A \tag{5.3}$$

$$plus\ A = A^{1.25} \tag{5.4}$$

The membership functions of the right-hand terms are computed using the following Equations:

$$\mu_{A^2}(x) = \mu_A{}^2(x) \tag{5.5}$$

$$\mu_{A^{1/2}}(x) = (\mu_A(x))^{1/2} \tag{5.6}$$

$$\mu_{1-A}(x) = 1 - \mu_A(x) \tag{5.7}$$

$$\mu_{A^{1.25}}(x) = (\mu_A(x))^{1.25} \tag{5.8}$$

The modifiers corresponding to "very" and "more or less" are also known as "concentration" and "dilation" respectively and are represented by CON and DIL. They are computed using Equations 5.9 and 5.10.

$$CON(A) = very\ A \tag{5.9}$$

$$DIL(A) = more\ or\ less\ A \tag{5.10}$$

In natural language, connectives and hedges are sometimes ambiguous and have no universal meaning. For instance, "and" may be as well logical as compensatory,

the "or" may be "exclusive" or "inclusive". "very" may indicate an increase in precision or a change in category. In fact, the human mind can perceive only a small number of categories.

5.4.1 Fuzzy Rules

One of the major components of a fuzzy logic system are *Rules*. The rules are expressed as logical implications. Fuzzy logic rules are "if-then" rules and define relations between linguistic values of outcome (then-part, i.e., consequent) and linguistic values of condition (if-part, i.e. antecedent). They are expressed as:

$$IF\ u_1\ is\ F_1^l\ and\ u_2\ is\ F_2^l\ and\ ...\ u_p\ is\ F_p^l\ THEN\ v\ is\ G^l$$

where l=1, 2, ... , M.

F_i^l and G^l are fuzzy sets. For example:

If *the delay is small and the area is medium* Then *the solution is accepted.*

Rules are a form of propositions. A *proposition* is an ordinary statement involving terms which have been defined, e.g., "The damping ratio is low". Consequently, we could have the following rule: "If the damping ratio is low, Then the system's impulse response oscillates a long time before it dies out" [18]. In traditional propositional logic an implication is said to be *true* if one of the following holds:

1. antecedent is true, consequent is true,

2. antecedent is false, consequent is false, and

3. antecedent is false, consequent is true.

The implication is called *false* when "antecedent is true and consequent is false". In fuzzy logic, a rule is fired as long as there is a nonzero degree of similarity between the first premise and the antecedent of the rule, and the result of such rule-firing is a consequent that has a nonzero degree of similarity to the rule's consequent. Because there can be lots of data, it is quite likely that there will be some conflicting rules, i.e, rules with the same antecedents but different consequents [18].

Rules may be provided by experts or can be extracted from numerical data. In either case, *engineering rules* are expressed as a collection of IF-THEN statements. Basically, building a rule needs the understanding of:

1. linguistic variables versus numerical values of a variable;
2. quantifying linguistic variables by using fuzzy membership functions;
3. logical connections for linguistic variables;
4. implications, i.e., "IF A THEN B";
5. additionally, combination of more than one rule.

5.4.2 Linguistic Variable

A linguistic variable, as defined in [49], is a variable whose values are words or sentences in a natural or artificial language. A linguistic variable is characterized by a quintuple (Ω, T(Ω), X, G, N), detailed as follows:

1. Ω is the name of the linguistic variable;

2. T(Ω) is the term-set of Ω, i.e., the collection of its linguistic values;

3. X is a universe of discourse;

4. G is a syntactic rule which generates the terms in T(Ω); and

5. N is a semantic rule which associates with each linguistic value its meaning,

N(ω) denotes a fuzzy subset of X for each $\omega \in$ T(Ω). The following example from [14] will help clarify the meaning of a linguistic variable. Let X be the universe of *timing delay*, A be the fuzzy set *timing delay near 5ns*, and $\mu_A(.)$ be the membership function for A. In this example, *timing delay* is a linguistic variable (Ω=*timing delay*). The linguistic values of *timing delay* can be defined as T(Ω)={ *very small delay, somewhat small delay, small delay, large delay, very large delay, non-critical, critical* }. The universe of discourse X is a possible range of {*timing delay*} for some designs. N(ω) defines a fuzzy set for each linguistic value $\omega \in$ T(Ω). The term-set of a linguistic variable is the collection of all its linguistic values. The meaning of a linguistic value x is characterized by a compatibility function, C: $\rightarrow$ [0, 1] where [0, 1] is the universe of discourse.

In the case of the linguistic variable "Area", the numerical variable *area* whose values are numbers constitutes the *base variable* for "Area". The compatibilities of the numerical areas 4, 5, 6 with the *fuzzy restriction* labelled "*Medium*" might be 1, 0.5, and 0 respectively as would be represented by the graph of Figure 5.3.

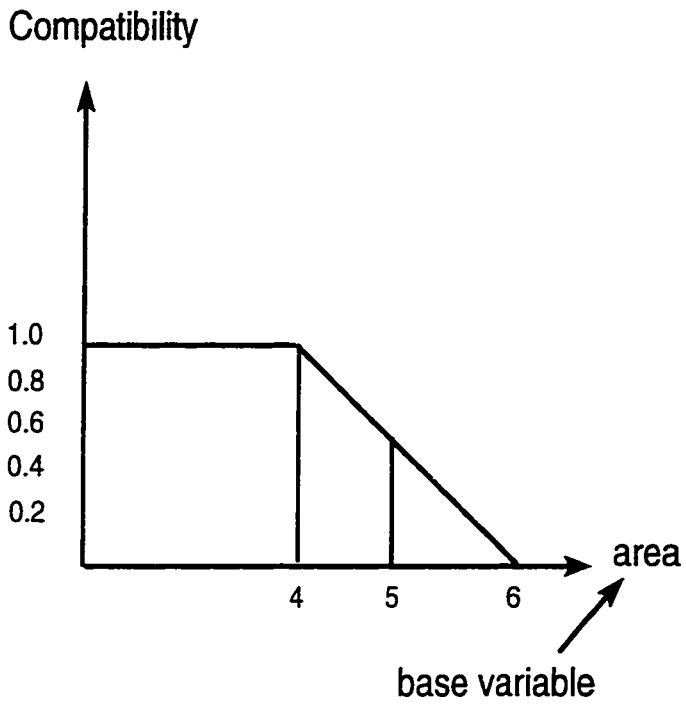


Figure 5.3: Compatibility function versus base variable "area".

5.4.3 Membership Functions

The membership function is supposed to be a good model of the way people perceive categories [10]. Category membership is not always a yes-or-no matter, but rather a matter of degree. A membership function is generally not absolutely defined and is fuzzy in the sense that as soon as it has a good shape, it can be considered a satisfactory approximation. In [51], a number of ways are used to study the membership assignment problem. First, a numerical measurement scale can be established. The construction of a numerical scale for membership values is performed in such a way that the scale satisfies some conditions imposed on a

rational measurement system. Second, excessive subjectiveness of an individual is reduced by incorporating the expertise of a number of individuals. Third, Zadeh introduced fuzzy linguistic variables to evaluate the memberships of complicated actions. A reason to study the membership assignment problem is to eliminate excessive arbitrariness in the qualitative preference structure due to idiosyncracy, because no meaningful decision can be made in a very chaotic situation.

Definition : let X be a space of points representing objects and let x be an element of X. A *fuzzy set* A in a universe of discourse X is characterized by a *membership* function $\mu_A(x)$ which associates with each point in X a real number in the interval [0, 1], with the value of $\mu_A(x)$ at x representing the *grade of membership* of x in A as shown in Figure 5.4 [47].

As $\mu_A(x)$ approaches 1, the grade of membership of x in A becomes high. In the ordinary sense of a set, $\mu_A(x)$ can be either 1 or 0 according to whether x is in A or not. Membership functions, $\mu_A(x)$, are for the most part, associated with terms that appear in the antecedents or consequents of rules, or in phrases. The most commonly used shapes for membership functions are triangular, trapezoidal, piecewise linear, and Gaussian.

5.4.4 Operations With Fuzzy Sets

Fuzzy logic operators belong to two categories: "non-compensatory" where the weaker element cannot compensate for the stronger element and "compensatory"

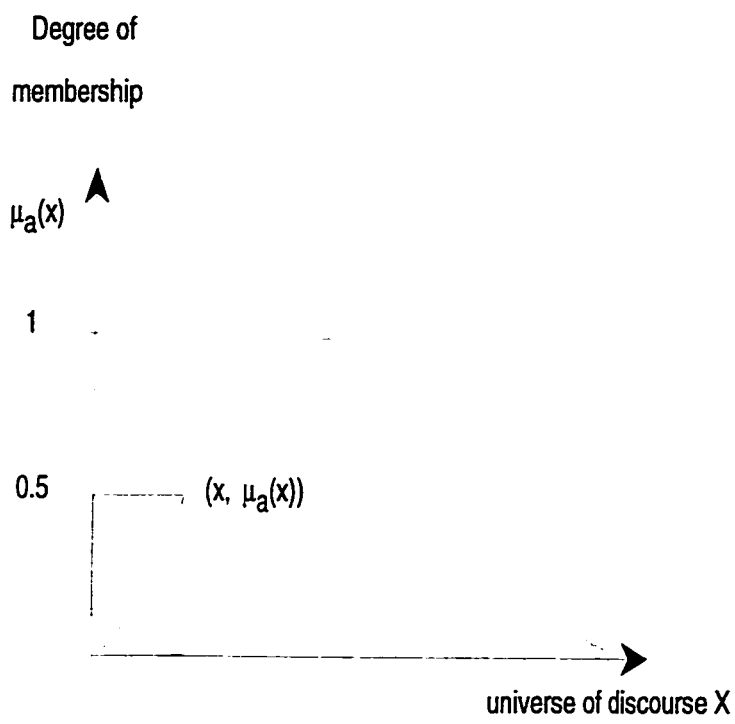


Figure 5.4: Membership function for fuzzy set A.

where the weaker element influences the result of the operator.

(a) Non-Compensatory Operators

There are many non-compensatory operators [48]:

Empty fuzzy set: A fuzzy set is empty if and only if its membership function is identically zero on X.

Equality: Two fuzzy sets A and B are equal (A=B), if and only if $\mu_A(x)=\mu_B(x)$ for all $x \in$ X.

Complementarity: The complement of a fuzzy set A is denoted by A' and is defined by: $\mu_{A'}=1-\mu_A$.

Containment: A is contained in B (A $\subset$ B or A$\subseteq$B) if and only if: $\mu_A \leq \mu_B$.

Union: The union of a fuzzy set A and a fuzzy set B is a fuzzy set C (C = A$\cup$B) whose membership function is: $\mu_C(x)$=Max($\mu_A(x)$, $\mu_B(x)$), such that $x \in X$.

Intersection: The intersection of fuzzy set A with fuzzy set B is a fuzzy set C (C = A$\cap$B) whose membership function is: $\mu_C(x)$=Min($\mu_A(x)$, $\mu_B(x)$), $x \in X$.

Algebraic product: The algebraic product of fuzzy set A and fuzzy set B (AB) is defined by: $\mu_{AB}=\mu_A \times \mu_B$.

Algebraic sum: The algebraic sum of fuzzy set A and fuzzy set B (A+B) is defined by: $\mu_{A+B}=\mu_A+\mu_B$.

Absolute difference: The absolute difference of A and B ($|A - B|$) is defined by: $\mu_{|A-B|}=|\mu_A - \mu_B|$.

(b) Probabilistic-like Operators

Probabilistic-like operators for both the intersection and the union are defined as follows [10]:

Intersection: $\forall x \in X,\ \mu_{A.B}(x) = \mu_A(x).\mu_B(x)$

Union: $\forall x \in X,\ \mu_{A\dot{+}B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x).\mu_B(x)$

Such operators reflect a trade-off between A and B, and are said to be *interactive*, as opposed to min and max.

(c) Compensatory Operators

The following are the compensatory parameterized operators for the intersection and union as proposed by Dubois and Prade in [10]:

The intersection of two fuzzy sets A and B is defined as:

$A \cap B = \{(x, \mu_{A \cap B})(x)) \mid x \in X\}$

where,

$$\mu_{A \cap B}(x) = \frac{\mu_A(x).\mu_B(x)}{\max\{\mu_A(x), \mu_B(x), \alpha\}} \tag{5.11}$$

$\alpha \in [0, 1]$.

The intersection operator is decreasing with respect to α and lies between min $\{\mu_A(x), \mu_B(x)\}$.

The union of two fuzzy sets A and B is defined as:

$A \cup B = \{(x, \mu_{A \cup B}(x)) | x \in X\}$

where,

$$\mu_{A \cup B}(x) = \frac{\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x) - \min\{\mu_A(x), \mu B(x), (1-\alpha)\}}{\max\{(1-\mu_A(x)), (1-\mu_B(x)), \alpha\}} \tag{5.12}$$

In a multi-objective optimization, it is important to make use of all the information of every criterion. In this case, it is recommended that interactive or compensatory versions of the AND and OR operators be used.

5.5 Multiobjective optimization

In order to solve multiobjective problems, the cost function should reflect all the objectives. Multiple objectives used to be combined into a scalar objective function, usually through a linear combination (weighted sum) of the multiple attributes. The cost of a solution x is expressed as:

$$f(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) + \cdots + w_n \cdot f_n(x) \tag{5.13}$$

where n is the number of objective functions. $f(x)$ is a combined objective function, $f_i(x)$ is the ith objective, and w_i is the weight of the ith objective. In the floorplan design problem, the cost function consists of three terms:

1. the area of floorplan bounding rectangle,
2. the overall interconnection wire-length, and
3. its timing performance.

Since the three terms represent different quantities, they are normalized to fall in the same range. Then, depending on the designers preferences, different weights are assigned to each term. Objectives of higher preferences are given higher weight.

Pareto Optimality: A notion of optimality that respects the integrity of each of the separate criteria is the concept of Pareto optimality. Here, suppose we wish to

minimize two objectives, expressed as f_1 and f_2. Let A, B, C, D, E, and F, be six possible solutions to our optimization problem, with the following costs:

$$A : (10, 90) \quad B : (20, 70) \quad C : (08, 75)$$

$$D : (15, 60) \quad E : (09, 65) \quad F : (14, 63)$$

That is, solution A has a value of f_1=10 and f_2=90. If we plot the 6 points f_1 versus f_2, obviously those that are lower and on the left are regarded as the best. Points C and D are good choices since there are no points better than these in both the criteria. C is best with respect to f_1 and D with respect to f_2. On the other hand, A and B are poor choices. Solution A(10,90) is dominated by solution C(08,75), since $10 > 8$ and $90 > 75$. If any solution p is to the right and top of another solution q, then we say p is dominated by q. A is also dominated by E. Similarly, B(20,70) is dominated by D(15,60), E(09,65) and F(15,60). The set of solutions that are not dominated by any other solution is {C, D, E, F}. In this problem, as in any other multiobjective optimization problem, such a set of solutions comprises the Pareto-optimal (P-optimal) set. It is from this set that the decision maker has to make a choice. The Pareto optimality concept does not assist in making a *single* choice.

VEGA: One of the first works in applying GAs to multiobjective optimization problems was by Schaffer [31]. Schaffer suggested a Vector Evaluated Genetic Algorithm (VEGA) for finding Pareto optimal solutions. In VEGA, the population is divided into equally sized, disjoint sub-populations, each governed by a different

objective function. Selection is performed independently of each criterion; however crossover is performed across sub-population boundaries. The problem with this scheme is that, independent selection of best solution in each criterion results in potential bias against middle solutions (such as E and F in our case). That is, those which are good but not the best with respect to any single criterion. VEGA mostly finds extreme solutions on the Pareto front. Schaffer suggested two approaches to improve VEGA. One is to provide a heuristic selection preference for non-dominated individuals in each generation. The other is a cross-breeding among the "species" by adding some mate selection.

MOGA: Recently, Murata and Ishibuchi proposed a Multiobjective GA (MOGA) [19] which uses a weighted sum of multiple objective functions to combine them into a scalar fitness function. The key feature of MOGA is that the weights attached to the multiple objective functions are *not* constant but randomly specified for each selection. Therefore, the direction of search in MOGA is not fixed. Weights are chosen as follows 5.14:

$$w_i = \frac{random_i(\cdot)}{\sum_{j=1}^{n} random_j(\cdot)} \tag{5.14}$$

where $random_j(\cdot)$ is a non-negative uniformly selected random number. During the execution of MOGA, a tentative set of Pareto optimal solutions is stored and updated at every generation. A certain number (say N_{elite}) of individuals are randomly selected from the set at each generation. These solutions are used as

elite individuals in MOGA. This elite-preserve strategy has the effect of keeping a good variety within each population.

5.6 Conclusion

Genetic algorithm, simulated annealing, and tabu search heuristics are used for optimization of problems which may consist of multiple objectives. In a multi-objective optimization problem, it is usually impossible to optimize all functions defined for each objective of the problem simultaneously. The optimum solution is viewed in terms of a utility function and constitutes the best compromise solution for different objectives that are merged into one. Hence, preference parameters have to be specified in order to solve the problem. Changing these preference parameters may lead to another best compromise solution. When using a utility function to merge multiple objectives, there is no clear way as to how to provide the physical interpretation. In this situation, fuzzy set theory can be used to overcome this problem. A fuzzy rule shows a clear relationship between the objectives and the cost.

Chapter 6

Floorplan Cost Evaluation Using Fuzzy Logic

6.1 Introduction

A fuzzy logic rule expresses the way the linguistic variables (objectives) are interrelated and the relationship between these objectives and the cost. For the floorplanning problem with three objectives to be optimized, namely area, length, and delay, the goal is to find a good solution which corresponds to a "good floorplan". A good floorplan consists of a small area, short length, and/or low delay as shown in Figure 6.1.

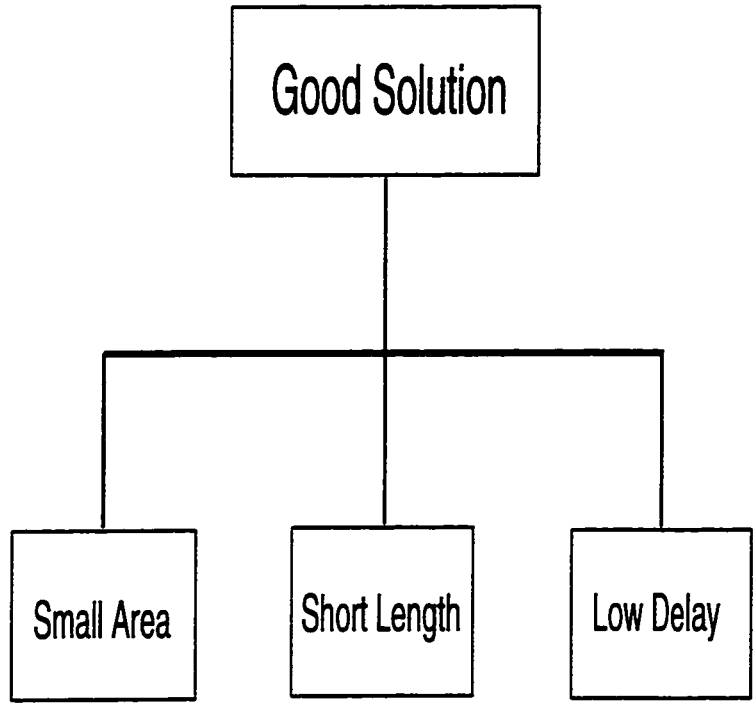


Figure 6.1: Three basic components for a good floorplan.

6.2 Fuzzy Rules

The three criteria, small area, short length, and low delay are combined in a number of ways using fuzzy operators to generate rules with a single consequent, i.e., "good solution." For example:

- R1: IF (area is small and length is short and delay is low) THEN solution is good.
- R2: IF (area is small or length is short) and delay is low THEN solution is good.
- R3: IF area is small and (delay is low or length is short) THEN solution is good.

Since it is impossible to get the best value for every objective during the process of optimization, we would like to get a solution (floorplan) where there is a good compromise between the area and the length or delay. In other words, our emphasis is to obtain the smallest area provided that solution has short length or low delay. In this work, we adopted fuzzy rule R3.

6.2.1 Preference Rules

In addition to fuzzy logic rules, an additional set of rules that give preferences to one criterion or another may be used to emphasize or de-emphasize a criterion (or criteria) involved in making decision during the search. As an example, the following preference rules can be considered:

- PR1: small area has a strong preference over low delay.
- PR2: small area has a strong preference over short length.
- PR3: small area has a strong preference over low delay and short length.
- PR4: short length has strong preference over low delay.

The word "preference" is viewed as a linguistic variable with many possible linguistic values such as "strong", "mild", "weak", etc. Our interest goes to "strong preference" whose membership function $p(.)$ is defined by a constant value. In this work, we adopted the preference rule PR3.

6.3 Membership Functions

The fuzzy logic rule (i.e., rule R3) has three linguistic variables. One linguistic value is assigned to each linguistic variable. A membership function is associated with each linguistic value. Next, we discuss the membership function associated with each linguistic value.

6.3.1 Area Membership Function

First, we determine two extreme values for the area, i.e., minimum value and maximum value. The minimum value, "area_min" is found by computing the sum of the areas of the basic blocks using Expression 6.1 shown below.

$$\text{“}MinArea\text{”} = \sum_{i=1}^{N} a_i \tag{6.1}$$

where N is the number of basic blocks and a_i is the area of basic block i. On the other hand, the maximum value for the area is found by computing the floorplan area of each individual (floorplan representation) in the initial generation. The largest area is the one that will be considered as the maximum area, i.e. "MaxArea". In this work, we normalized the value of area with respect to "MaxArea". The shape of the membership function for the linguistic value "small area" is shown in Figure 6.2. The membership value for the normalized area is

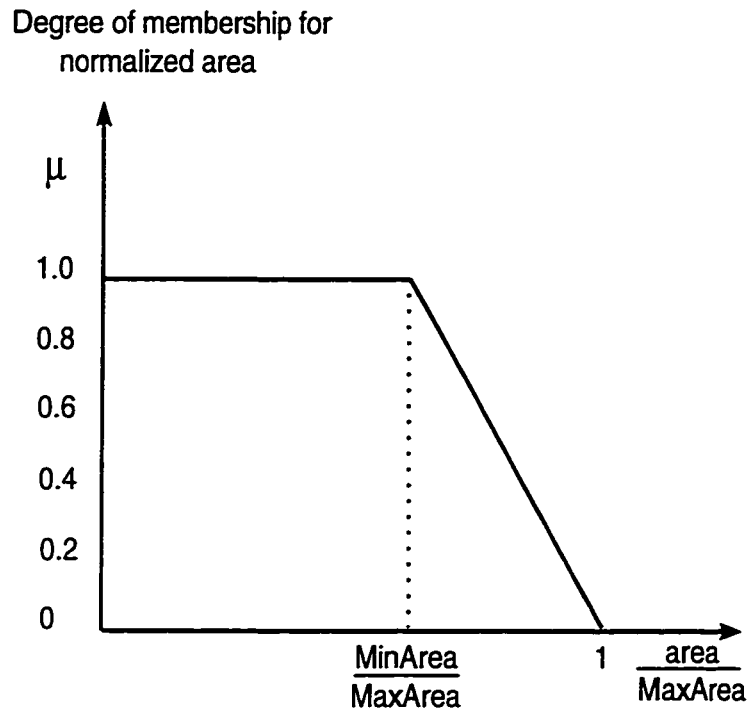


Figure 6.2: Membership function for normalized "small area".

then computed using Equation 6.2:

$$\mu_A(area) = \begin{cases} 1 & if \quad \frac{area}{area\_max} \leq \frac{area\_min}{area\_max} \\ \frac{\frac{area}{area\_max} - 1}{\frac{area\_min}{area\_max} - 1} & if \quad \frac{area\_min}{area\_max} < \frac{area}{area\_max} < 1 \\ 0 & if \quad \frac{area}{area\_max} \geq 1 \end{cases} \tag{6.2}$$

6.3.2 Length Membership Function

The maximum length is determined from the initial generation. The interconnect length between the basic blocks is computed for each solution. The largest length within the initial generation of floorplans is retained as the maximum one, i.e., "MaxLength." The minimum length, i.e. "MinLength", is computed using an

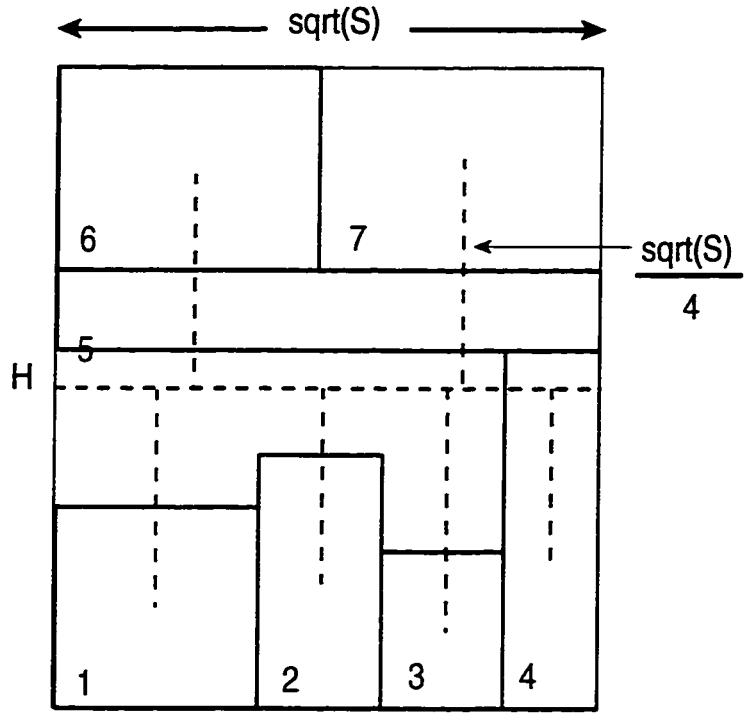


Figure 6.3: Approximation of the minimum wire length.

approximation technique. There is no relation between the initial generation of floorplan solutions and the minimum length. The computation of this minimum length depends on the netlist only. We assume that the modules connected by a particular net are packed in a block that is squarish and whose area is S. As shown in Figure 6.3, the horizontal line H has a length equal to $\sqrt{(S)}$. A connection from a module to the horizontal line H is approximated by $\frac{1}{4}\sqrt{(S)}$. Therefore, an estimate of the overall netlength would be $(1+\frac{k}{4})\times\sqrt{S}$. The approximation technique is illustrated in Figure 6.4. Normalizing the length with respect to "MaxLength" leads to the membership function having the shape of Figure 6.5. The membership value for the normalized length is obtained with Equation 6.3:

```
Procedure length_min
        For each net i of the netlist;
            IF k=2
                Compute the sum S of areas of basic blocks in net i;
                The minimum estimated length for net i is √S;
            ELSE
                Compute the sum S of areas of basic blocks in net i;
                The minimum estimated length for net i is:
                (1+k/4)×√S;
            END IF
            Accumulate the estimated minimum length for each net i
            of the netlist.
        End For
End Procedure
```

Figure 6.4: Minimum length computation algorithm.

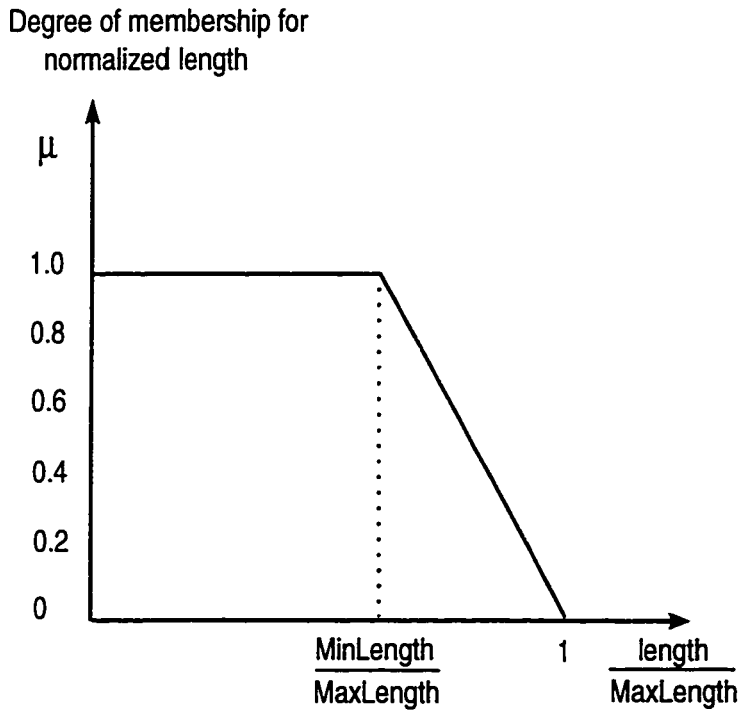


Figure 6.5: Membership function for normalized "short length".

$$\mu_L(length) = \begin{cases} 1 & if \frac{length}{length\_max} \leq \frac{length\_min}{length\_max} \\ \frac{\frac{length}{length\_max}-1}{\frac{length\_min}{length\_max}-1} & if \frac{length\_min}{length\_max} < \frac{length}{length\_max} < 1 \\ 0 & if \frac{length}{length\_max} \geq 1 \end{cases} \quad (6.3)$$

6.3.3 Delay Membership Function

The maximum delay is computed from the initial generation. The delay for each initial individual is determined. The largest value for the delay within the initial generation is retained as the maximum delay, i.e, "MaxDelay." The computation of the minimum delay or "MinDelay" relies on the computation of the minimum length. We need to consider two types of connections, horizontal and vertical denoted by h and v respectively, where h=v=$\frac{"MinLength"}{2}$. With these two values, the computation of the minimum interconnect delay for each net i is accomplished as follows using Equations 6.4 to 6.7:

$$Area\_Capacitance_i = (C_{am1} \times v_i + C_{am2} \times h_i) \times w \quad (6.4)$$

$$Fringe\_Capacitance_i = 2 \times ((w + v_i) \times C_{fm1} + (w + h_i) \times C_{fm2}) \quad (6.5)$$

$$C_i = Area\_Capacitance_i + Fringe\_capacitance_i \quad (6.6)$$

$$MinimumInterconnectDelay = LF_i \times C_i \quad (6.7)$$

where

v_i and h_i are vertical and horizontal length of net i.

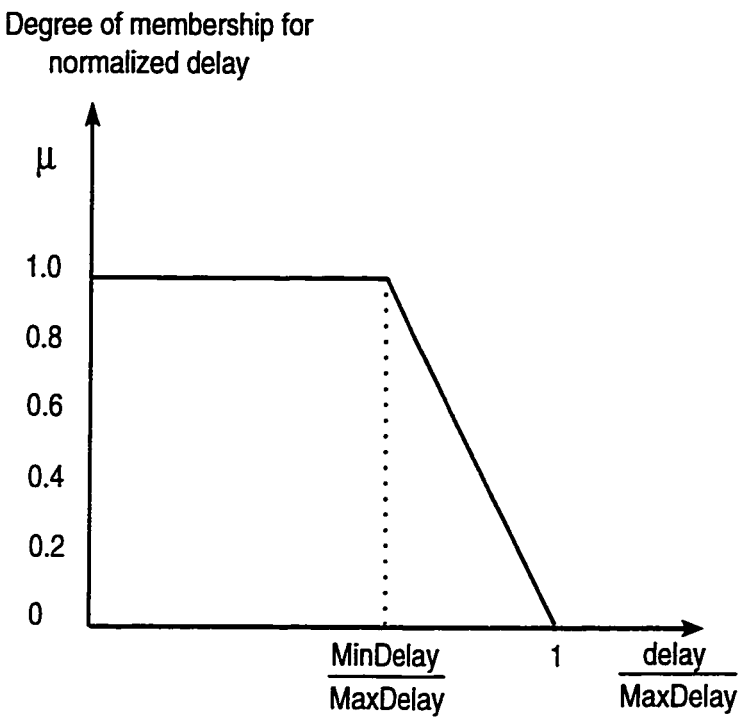


Figure 6.6: Membership function for normalized "low delay".

w = the wire width.

LF_i = load factor of the input pin driving net i. C_{am1} = Plate capacitance per Area of metal m_1.

C_{fm1} = Fringe capacitance per Length of metal m_1.

C_{am2} = Plate capacitance per Area of metal m_2.

C_{fm2} = Fringe capacitance per Length of metal m_2.

After normalizing the delay with respect to "MaxDelay", the shape of the membership function becomes as shown in Figure 6.6. The membership value for the

normalized delay is computed using expression 6.8 shown below:

$$\mu_D(delay) = \begin{cases} 1 & if \frac{delay}{delay\_max} \leq \frac{delay\_min}{delay\_max} \\ \frac{\frac{delay}{delay\_max} - 1}{\frac{delay\_min}{delay\_max} - 1} & if \frac{delay\_min}{delay\_max} < \frac{delay}{delay\_max} < 1 \\ 0 & if \frac{delay}{delay\_max} \geq 1 \end{cases} \quad (6.8)$$

6.4 Cost Computation

For the floorplanning problem, the three objectives considered are area, wirelength, and delay. In the weighted sum implementation of the cost function, the three objectives are normalized with respect to their upper bound and combined as a weighted sum. When using Fuzzy Logic, the expression giving the cost function is derived from fuzzy rules which are built using linguistic variables (area, length, and delay). These linguistic variables are combined with fuzzy operators commonly known as connectives in the natural language, i.e., "and" and "or". In this work, a number of approaches and models of fuzzy operators and rules have been used to derive a cost function. Fuzzy algebra allows the move from the linguistic domain to the numerical domain. The quality of solution is specified in linguistic terms expressed in the form of Fuzzy rules and preference rules.

6.4.1 Non-compensatory Operators

With the non-compensatory operators, the application of the fuzzy rule R3 gives the following cost function:

$$Cost(x) = \min(\mu_{area}(x), \max(\mu_{length}(x), \mu_{delay}(x))) \tag{6.9}$$

Introducing preference rule PR3 where $p(.)$ is the membership function for the linguistic value "strong preference", the cost function becomes as formulated by Equation 6.10:

$$Cost(x) = \min(p(.)\mu_{area}(x), (1 - p(.)) \max(\mu_{length}(x), \mu_{delay}(x))) \tag{6.10}$$

6.4.2 Compensatory Operators

The cost function derived from the fuzzy rule R3 with compensatory operators of Dubois and Prade is formulated by Equation 6.11:

$$Cost(x) = \frac{\mu_{area}(x)(\mu_{length \cup delay}(x))}{\max\{\mu_{area}(x), \mu_{length \cup delay}(x), \alpha\}} \tag{6.11}$$

where,

$$\mu_{length \cup delay}(x) = \frac{\mu_{length}(x) + \mu_{delay}(x) - \mu_{length}(x)\mu_{delay}(x) - min\{\mu_{length}(x), \mu_{delay}(x), (1 - \alpha}{max\{(1 - \mu_{length}(x)), (1 - \mu_{delay}(x)), \alpha\}} \tag{6.12}$$

With the preference rule PR3, the expression of the cost function becomes as formulated by Equation 6.13:

$$Cost(x) = \frac{p(.)\mu_{area}(x).(1-p(.))\mu_{lengthUdelay}(x)}{\max\{p(.)\mu_{area}(x).(1-p(.))\mu_{lengthUdelay}(x).\alpha\}} \tag{6.13}$$

where $p(.)$ is the membership function for the linguistic value "strong preference".

6.4.3 Probabilistic-like Operators

With the probabilistic-like operators, the cost function derived from the same above mentioned rule, i.e., rule R3 is defined by Equation 6.14:

$$Cost(x) = \mu_{area}(x)(\mu_{length}(x) + \mu_{delay}(x) - \mu_{length}(x).\mu_{delay}(x)) \tag{6.14}$$

In fact, the probabilistic-like operators are a particular case of the compensatory operators, i.e, when α=1. If one is to give more emphasis on particular criterion, he can resort to hedges. For example one might be looking for floorplans with more or less small area or very short length or delay, he would be using a rule as below:

If (area is "more or less" small and (length is "very" short or delay is "very" low)) Then good solution

The hedge "more or less" causes a dilation of the target criterion while the hedge "very" is interpreted as a concentration of the desired criterion. The expression of

the cost function will be computed as shown below:

$$Cost(x) = \mu_{area}(x)^{\frac{1}{2}}(\mu_{length}(x)^2 + \mu_{delay}(x)^2 - \mu_{length}(x)^2.\mu_{delay}(x)^2) \qquad (6.15)$$

6.4.4 Additive Combiner

A number of experiments were conducted using the additive combiner approach [7]. The following three rules were used simultaneously:

- R1: If area is small Then good solution.
- R2: If length is short Then good solution.
- R3: If delay is small Then good solution.

With these three rules, the cost function is a linear combination of weighted membership values for each objective, i.e, area, length, and delay, as shown in Figure 6.7. The expression giving the value for the cost is formulated by Equation 6.16:

$$Cost(x) = w1 \times \mu_{R1} + w2 \times \mu_{R2} + w3 \times \mu_{R3} \qquad (6.16)$$

where $w1$, $w2$, $w3$ are confidence factors (or degree of beliefs) associated with fuzzy rules R1, R2, R3 respectively. Applying hedges such as "more or less" and "very" to the three rules results in the following:

- R1: If area is "more or less" small Then good solution.
- R2: If length is "very" short Then good solution.

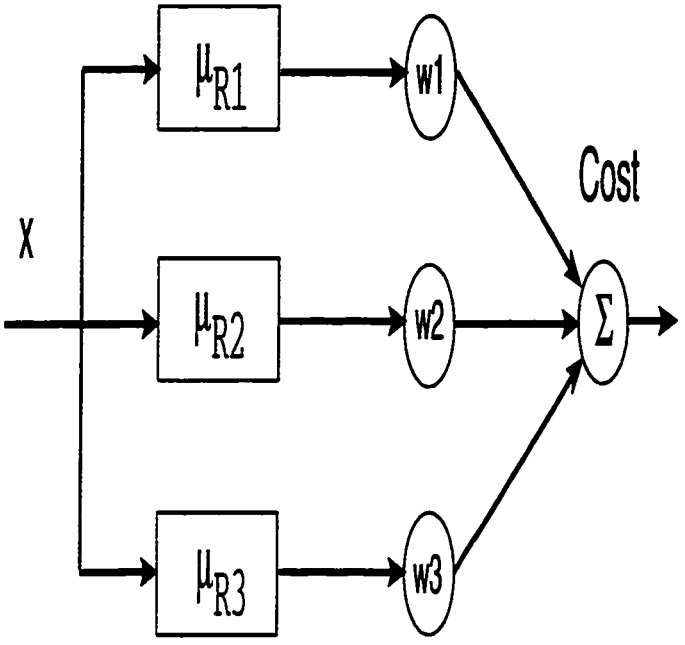


Figure 6.7: Additive combiner producing a fuzzy output as a combination of other fuzzy rule outcomes (here 3 rules).

- R3: If delay is "very" low Then good solution.

The value obtained from the cost function derived from the modified rule is computed using Equation 6.17:

$$Cost(x) = w1 \times \mu_{R1}^{\frac{1}{2}} + w2 \times \mu_{R2}^{2} + w3 \times \mu_{R3}^{2} \tag{6.17}$$

6.4.5 Ordered Weighted Averaging

Another rule that is suggested is the following:

Rule: IF (area is small or length is short or delay is low) THEN solution is good

where the "or" is implemented using the Ordered Weighted Averaging (OWA) *orlike* operator suggested by Yager [44] in the context of multicriteria decision making. The *orlike* and *andlike* operators are defined as follows: to the OWA:

1. OR-like soft

$$Cost = (1-\beta)\frac{1}{3}(\mu_{area} + \mu_{length} + \mu_{delay}) + \beta \max(\mu_{area}, \mu_{length}, \mu_{delay}) \quad (6.18)$$

2. AND-like soft

$$Cost = (1-\beta)\frac{1}{3}(\mu_{area} + \mu_{length} + \mu_{delay}) + \beta \min(\mu_{area}, \mu_{length}, \mu_{delay}) \quad (6.19)$$

Examples of hedges we will be using are "more or less" and "very".

Rule: IF (area is "more or less" small or length is "very" short or delay is "very" low) THEN good solution

Using this fuzzy rule, the fuzzy cost of a good floorplan solution would be computed as follows using OR-like soft operators:

$$Cost = (1-\beta)\frac{1}{3}(\mu_{area}^{\frac{1}{2}} + \mu_{length}^{2} + \mu_{delay}^{2}) + \beta \max(\mu_{area}^{\frac{1}{2}}, \mu_{length}^{2}, \mu_{delay}^{2}) \quad (6.20)$$

Using AND-like soft operator, we get:

$$Cost = (1-\beta)\frac{1}{3}(\mu_{area}^{\frac{1}{2}} + \mu_{length}^{2} + \mu_{delay}^{2}) + \beta \min(\mu_{area}^{\frac{1}{2}}, \mu_{length}^{2}, \mu_{delay}^{2}) \quad (6.21)$$

The value of β varies between 0 and 1. We notice that if β=1, the computation of the fuzzy cost is the same as when using the non-compensatory operators. On the other hand, if β=0 the computation of the cost is the average of the three membership values. In all cases, the value of the fuzzy "cost" ranges in the interval [0, 1].

6.5 Conclusion

The cost function is derived from the application of a fuzzy logic rule. Several algebraic models of fuzzy operators are described. The purpose is to translate the cost function from the linguistic domain into the computation domain and see for which type of operators the output is better with respect to three objectives.

Chapter 7

Experimental Results

7.1 Introduction

In this chapter, we present the results obtained using the Fuzzy approach. These results are then compared against those of the WS method. The illustrated experimental results are produced by the three heuristics (GA, SA, and TS). The outcomes of Fuzzy GA (FGA), Fuzzy SA (FSA), and Fuzzy TS (FTS) are finally compared with respect to the three objective criteria.

7.2 Results using GA

The different operators and their types used in our GA are listed in Table 7.1. The GA parameters are given the following values: 5% for mutation probability, 80% for crossover probability, 30% for inversion probability. The number of generations

Operator	Type
mutation	Single operator inversion Swapping of two randomly selected operands Swapping of an operand and an operator
crossover	Block inheritance CO1 Slicing inheritance CO2 PMX
inversion	Full right-to-left or left-to-right shift of operands

Table 7.1: Different mutation and crossover schemes used in our GA.

Circuit	Size	Area		Length		Delay	
name	number of blocks	max	min	max	min	max	r
highway	45	189184.00	88160.00	24570.54	4430.31	47.15	1(
add4	39	162221.87	90480.00	49826.64	4283.86	57.58	1
par	15	52326.00	34800.00	8161.84	960.16	66.68	3
par2	30	79864.33	40832.00	23106.22	2325.76	104.07	2
chip	141	844480.00	211120.00	168059.69	10415.84	100.70	2(

Table 7.2: Area, length, and delay for each circuit.

was set to 2000 with 10 individuals per generation.

The genetic algorithms were implemented and tested on five different VLSI circuits. The characteristics of these circuits are summarized in Table 7.2. The smallest test circuit has 15 basic blocks and the largest has 141 basic blocks. The initial generation is built through the following two steps:

1. For each individual, generate a list of operands corresponding to each basic block in a random sequence.

2. Inject operators in the list of operands such that the outcome is a valid polish expression.

In order for the overall delay to have a more pronounced effect in the optimization process, since most of it is due to the switching delay, we have inflated the interconnect delay by a factor of 5, i.e.:

$$\text{delay} = 5\times \text{ interconnect\_delay+switching\_delay.}$$

A number of experiments are performed using the WS approach and the Fuzzy approach to derive the cost function. The results are compared in order to justify the use of Fuzzy Logic in solving the problem. Different types of fuzzy operators are tested. The purpose is to determine which fuzzy operator leads to better outcomes in terms of solution quality. The flexibility introduced by the Fuzzy approach is mainly due to the fact that the problem is easily described in terms of natural language as Fuzzy rules. The preference of one objective over another can be specified by the use of modifiers.

7.2.1 Discussion of Results

The quality of solution produced with the Weighted Sum approach is very sensitive to the selection of relative weights for the three objective criteria. For example, increasing the weight for area does not always cause an improvement with respect to the floorplan area as illustrated in Table 7.3 [1]. A change to any of the weights usually causes an unpredictable change to the quality of solution. Therefore, it took quite a bit of efforts to identify the weight settings that gave best results. Results

[1] In Table 7.3 as well as in the coming ones, all entries are normalized with respect to their lower bound values

Table 7.3: Results of WS-GA for different weights on circuit "highway".

Relative Weights (w_1, w_2 w_3)	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
(0.8, 0.10,0.10)	1.30	3.54	2.10
(0.7, 0.15,0.15)	1.38	3.74	2.21
(0.6, 0.20,0.20)	1.36	3.64	2.12
(0.5, 0.25,0.25)	1.36	3.64	2.12
(0.4, 0.30,0.30)	1.31	3.50	2.20

Circuit	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
highway	1.36	3.64	2.12
add4	1.28	2.95	1.75
par	1.04	3.79	1.44
par2	1.18	3.05	1.82
chip	1.71	9.57	3.35

Table 7.4: Results of the weighted sum (WS-GA) approach obtained with the following values for the weights: $w1$=0.6, $w2$=0.2, $w3$=0.2

obtained using the weighted sum approach WS-GA are summarized in Table 7.4. To tune the FGA, we tried the different implementations of the fuzzy operators AND and OR, namely the "min" and "max" operators, the compensatory AND and OR operators, the probabilistic-like AND and OR as well as the additive combiner approach. Results are summarized in Table 7.5. According to these results, the probabilistic-like and the additive combiner approaches present better outcomes compared to both compensatory and non-compensatory operators in terms of area. For example, the best floorplan obtained for circuit "par" by the additive combiner approach has an area equal to 1.04 that of the smallest possible area, i.e., only 4% of dead space. For the same circuit, the floorplans obtained by

Circuit	*Approach*	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
highway	FGA-A	1.40	3.47	2.10
	FGA-B	1.37	3.51	2.25
	FGA-C	1.37	3.65	2.03
	FGA-D	1.34	3.71	2.11
add4	FGA-A	1.54	3.75	1.84
	FGA-B	1.47	3.51	1.87
	FGA-C	1.36	3.09	1.72
	FGA-D	1.33	3.30	1.77
par	FGA-A	1.45	3.54	1.41
	FGA-B	1.32	3.56	1.41
	FGA-C	1.09	3.57	1.41
	FGA-D	1.04	3.79	1.44
par2	FGA-A	1.41	2.87	1.77
	FGA-B	1.41	2.87	1.77
	FGA-C	1.26	2.96	1.82
	FGA-D	1.21	3.13	1.86
chip	FGA-A	2.29	10.27	3.00
	FGA-B	1.96	9.30	2.76
	FGA-C	1.67	8.54	2.98
	FGA-D	1.75	8.58	2.94

Table 7.5: Results obtained using FGA-A (*non − compensatory*), FGA-B (*compensatory*), FGA-C (*probabilistic − like*) and FGA-D (*additive combiner*) approaches respectively.

Circuit	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
highway	1.29	3.73	2.11
	1.38	3.47	1.97
add4	1.26	2.85	1.86
	1.53	2.91	1.84
par	1.04	3.79	1.43
	1.09	3.57	1.41
par2	1.18	3.05	1.83
	1.50	3.24	1.87
chip	1.85	8.91	3.09
	1.90	8.85	3.09

Table 7.6: FGA Results obtained using the OR-like soft and the AND-like soft operators respectively with β=0.6

the non-compensatory and the compensatory operators have 45 % and 32 % of dead space respectively.

In most of the test cases, the additive combiner approach outperforms the probabilistic-like operators except for the test case "chip" where the additive combiner leads to a floorplan with 75 % of dead space, where the amount of dead space resulting from the probabilistic-like approach is 67 %. On the other hand, we notice that the probabilistic-like operators give better results concerning the wirelength and delay than the additive combiner approach.

Results obtained using the AND-like and OR-like soft operators of the OWA approach with β=0.6 are illustrated in Table 7.6. The OR-like operator shows better performance compared to the AND-like operator with respect to the area, however, in test case "highway" the AND-like operator produced better results for the two other objectives (wirelength and delay). The optimal floorplan obtained

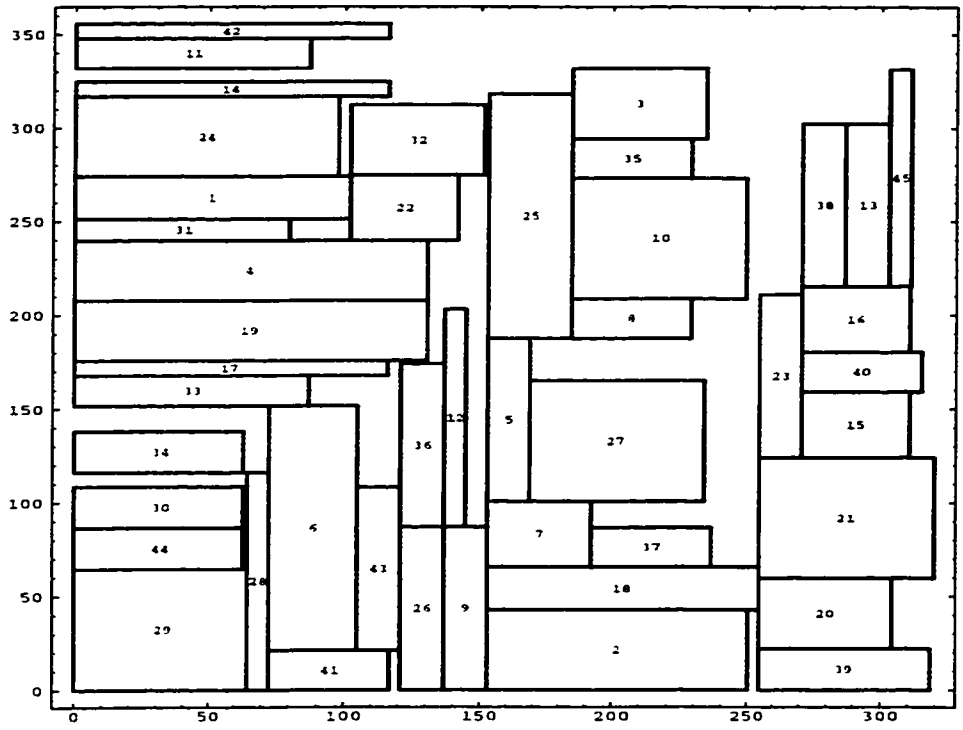


Figure 7.1: Best Floorplan solution of circuit "highway" obtained using FGA with the OR-like soft operator of the OWA approach where β=0.6 and no hedges applied to the rule.

using the OR-like soft operator is shown in Figure 7.1. Figures 7.2, 7.3, and 7.4 show the variations in the membership values of area, length, and delay respectively obtained for circuit "highway".

Figure 7.5 depicts the variation of the cost of the best solution (floorplan) with respect to the increase in the number of generations. Figure 7.6 shows the variations of the average cost throughout the generations.

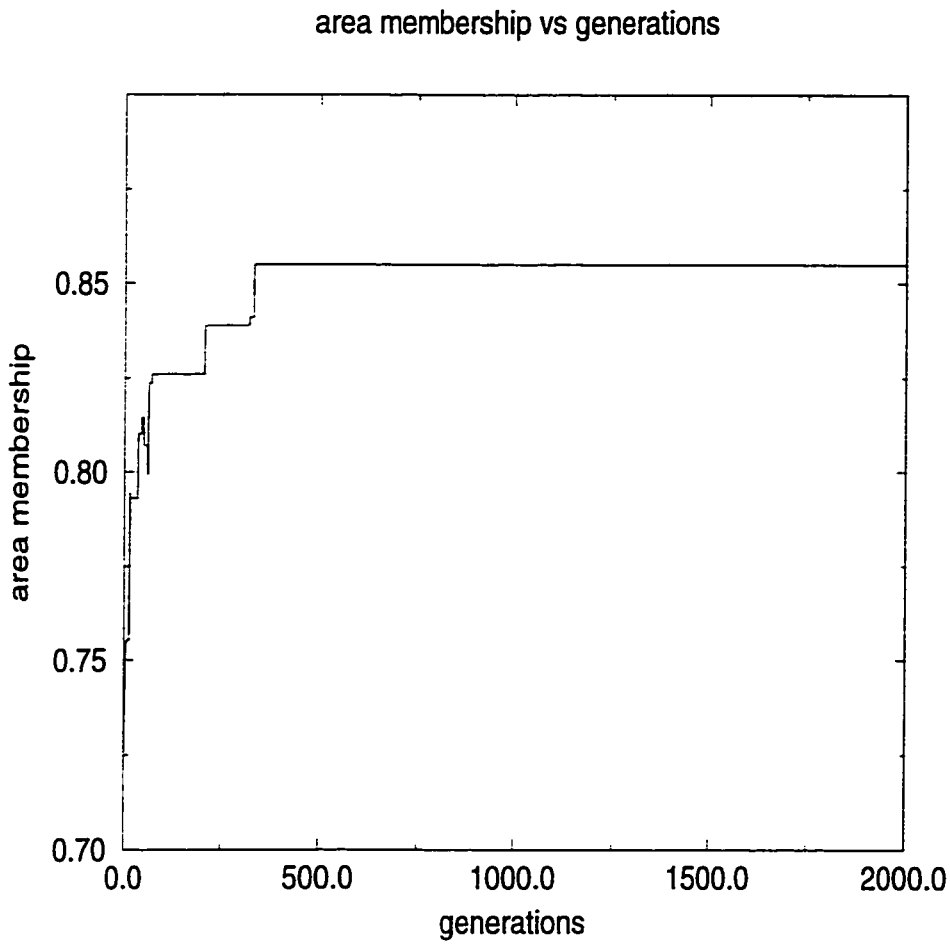


Figure 7.2: Area membership value vs generations plot of circuit "highway" obtained using FGA with the OR-like soft operator of the OWA approach where β=0.6 and no hedges applied to the rule.

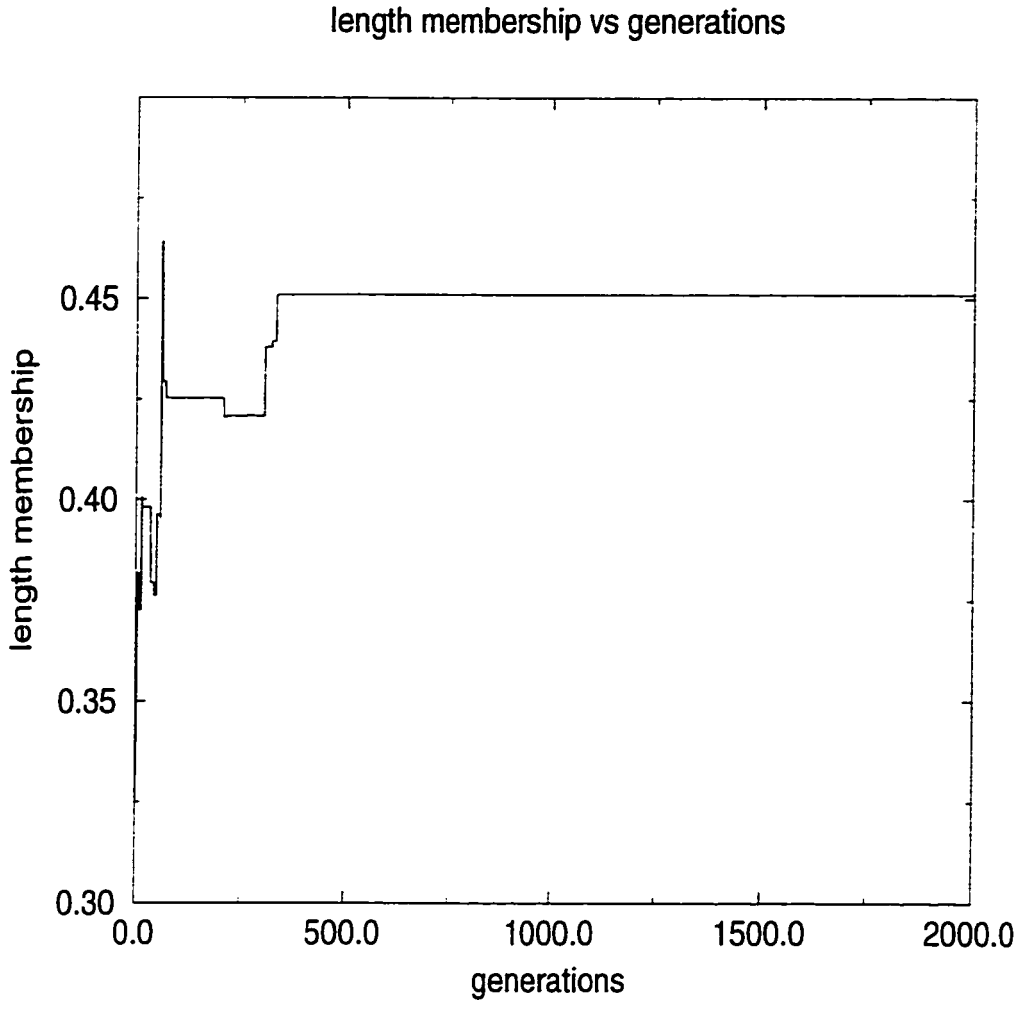


Figure 7.3: Length membership value vs generations plot of circuit "highway" obtained using FGA with the OR-like soft operator of the OWA approach where β=0.6 and no hedges applied to the rule.

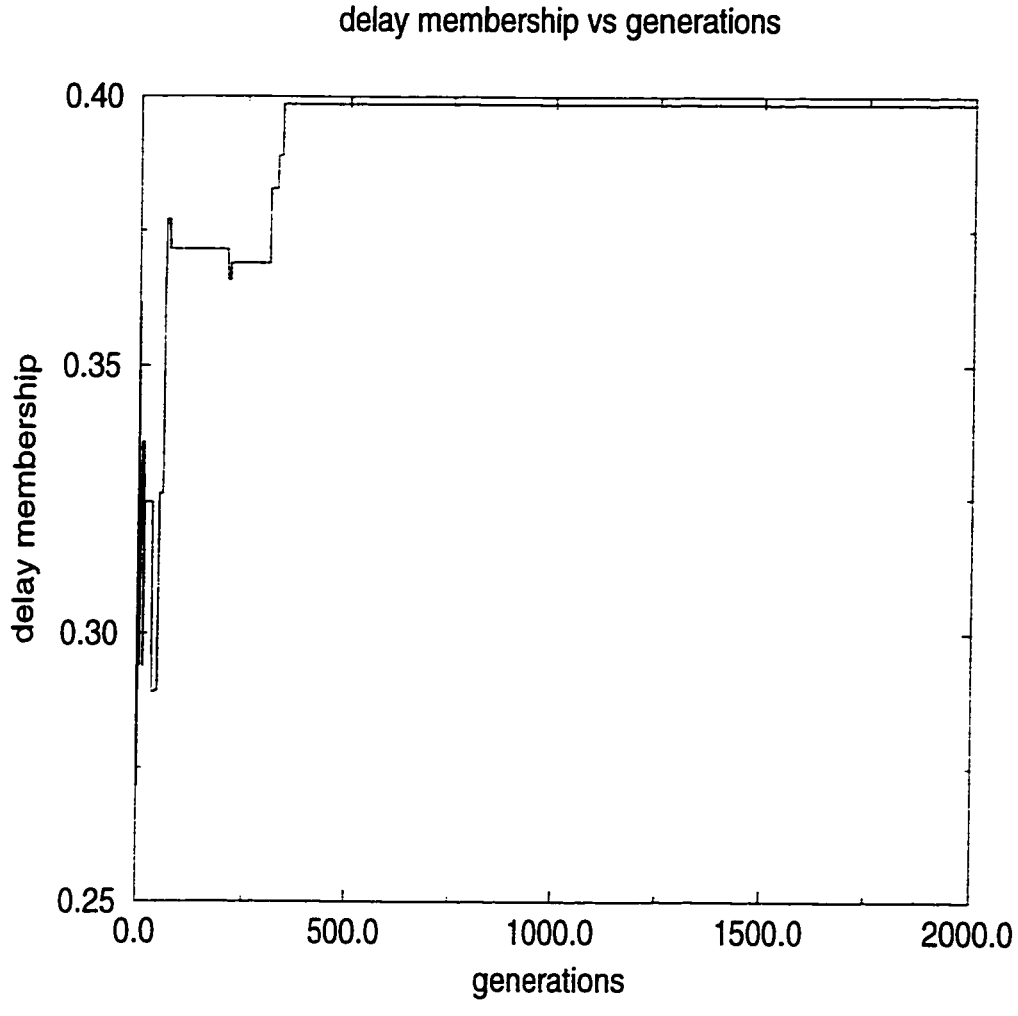


Figure 7.4: Delay membership value vs generations plot of circuit "highway" obtained using FGA with the OR-like soft operator of the OWA approach where β=0.6 and no hedges applied to the rule.

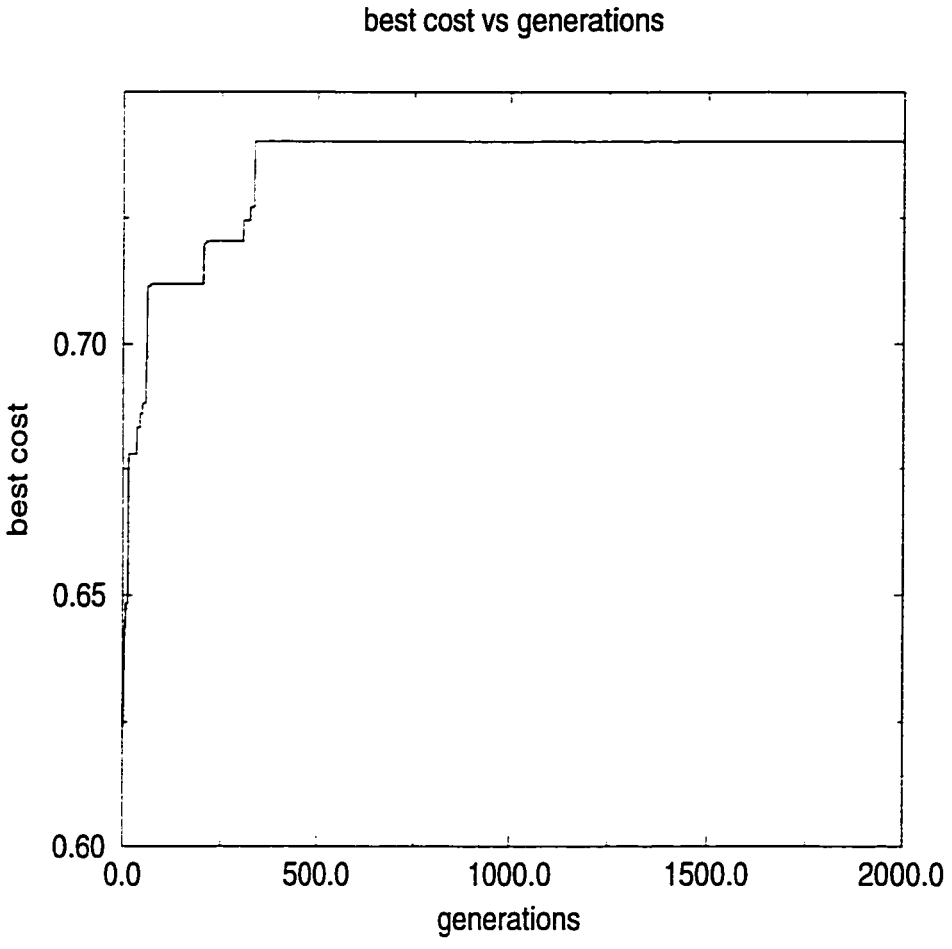


Figure 7.5: Best cost vs generations plot of circuit "highway" obtained using FGA with the OR-like soft operator of the OWA approach where β=0.6 and no hedges applied to the rule.

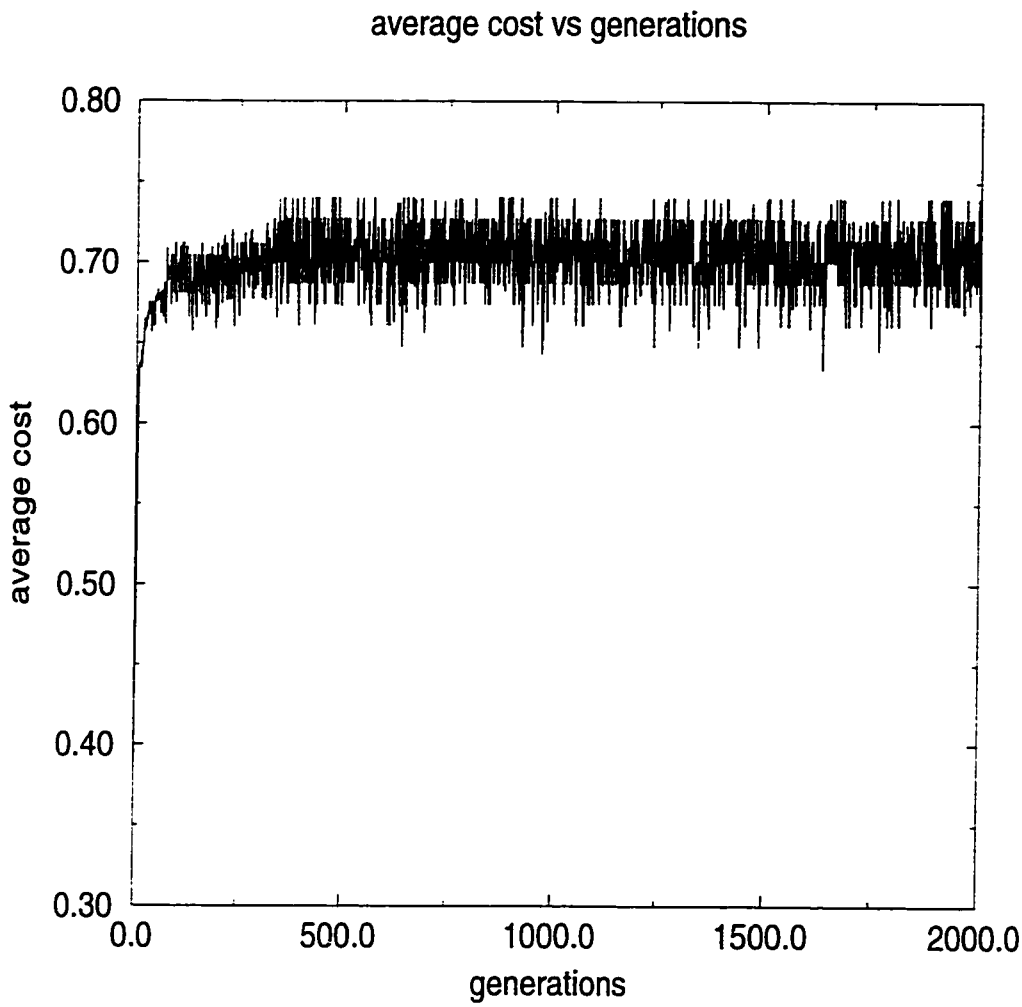


Figure 7.6: Average cost vs generations plot of circuit "highway" obtained using FGA with the OR-like soft operator of the OWA approach where β=0.6 and no hedges applied to the rule.

7.2.2 Effect of Hedges on the Quality of Results

A rule can be modified by either giving appropriate hedges to each linguitic value or by applying preference rules. A number of experiments have been conducted using the different approaches described above. The purpose of these experiments is to see how the results are affected by a modification of the rules.

Results summarized in Table 7.7 are obtained using non-compensatory and compensatory operators with preference rule PR3. The membership value of the linguistic value "strong preference" is assigned the value 0.8, i.e., $p(.)$=0.8. Comparing these results with those listed in Table 7.5 we notice that in most of the test cases the results are better in terms of area, wire length, and delay. For example, in circuit "add4", the non-compensatory operators lead to a floorplan with 32 % of dead space, however, the use of the same operators without any preference rule ends up with a dead space of 54 % of the minimum area, i.e, 40.7 % of improvement. The improvement is also recorded with respect to both length and delay by 26.93 % and 10.33 % respectively. The outcome produced by the compensatory operators shows 39 % of dead space. The amount of dead space is reduced when compared to the one obtained when no preference rule was applied (47 %). The amount of reduction is of the order of 17.02 %. The improvement is also recorded with respect to both length and delay, i.e., 5.13 % and 1.60 % respectively. We also notice that for some test cases, the quality of solutions obtained, does not

Circuit	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
highway	1.40	3.47	2.10
	1.40	3.34	2.07
add4	1.32	2.74	1.65
	1.39	3.33	1.84
par	1.46	3.55	1.41
	1.10	3.45	1.40
par2	1.42	2.87	1.78
	1.35	2.92	1.81
chip	1.92	9.40	2.74
	1.81	8.84	2.74

Table 7.7: FGA Results obtained using non-compensatory and compensatory approaches with preference rule PR3 where $p(.)$=0.8 and $\alpha = 0.5$.

improve, i.e., the solutions are approximately the same. This means that for some test cases, the use of the preference rule PR3 does not influence the quality of the solution for a particular objective value. For example, in test case "highway", the outcome of the GA using compensatory operators is a floorplan with 40 % of dead space. In Table 7.5, the amount of dead space is 37 %. There was no improvement towards reducing the dead space area. In both cases the results are approximately the same, about 2.2 % of difference between them. On the other hand, the use of the preference rule showed noticeable improvement for the other two objectives (length and delay) by a decrease in magnitude in the order of 4.84 % and 8.00 % respectively. Table 7.8 shows the results obtained using probabilistic-like and additive combiner approaches with hedges "more or less", "very", and "very" applied to area, length, and delay respectively. We observe that in most of the test cases, the results obtained using the same approaches but with no hedges (Table

Circuit	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
highway	1.42	3.75	2.12
	1.36	3.75	2.08
add4	1.34	3.05	1.76
	1.32	3.10	1.83
par	1.23	3.52	1.40
	1.18	3.68	1.43
par2	1.42	2.87	1.78
	1.21	3.13	1.86
chip	2.13	9.30	3.29
	1.92	8.98	3.06

Table 7.8: FGA Results obtained using probabilistic-like and additive combiner approaches with hedges "more or less", "very", "very", applied to area, wirelength, and delay respectively.

7.5) are better or approximately the same compared to those listed in Table 7.8. For example, in test case "chip", the floorplan obtained using the probabilistic-like operators has 67 % of dead space with no hedges applied in the fuzzy rule, but 113 % of dead space with the use of hedges, i.e., an increase in the order of 69 % of the overall dead space. For the two other objectives, the amount of increase is in the order of 8.9 % for the length and 10.40 % for the delay. In test case "par", the floorplan obtained using the GA with additive combiner approach shows 4 % of dead space area without the use of hedges and 18 % of dead space with hedges, i.e., an increase of about 13.46 % in dead space area. Both length and delay decrease by 2.9 % and 0.7 % respectively.

The outcome when using hedges with the OWA approach where β=0.6 is illustrated in Table 7.9. The hedges used are again; "more or less", "very", and "very" applied to area, wire length, and delay respectively. In most test cases,

Circuit	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
highway	1.28	3.66	2.10
	1.29	3.74	2.10
add4	1.42	3.33	1.91
	1.47	2.88	1.72
par	1.18	3.69	1.43
	1.09	3.57	1.41
par2	1.20	3.25	1.87
	1.35	2.92	1.81
chip	1.70	8.92	3.42
	2.06	9.00	3.05

Table 7.9: FGA Results obtained using the OR-like soft and the AND-like soft operators with β=0.6 and hedges "more or less", "very", "very" applied to area, wirelength, and delay respectively.

the results show a better quality with respect to area, length, and delay compared to those listed in Table 7.6 and obtained using the same approach but with no hedges applied in the fuzzy rule. For example, in test case "par2", the dead space area increases from 18 % to 20 % using OR-like operator but decreases from 50 % to 35 % with AND-like operator. For the two other objectives, length and delay, their values increase by an amount of 6.50 % and 2.18 % respectively with OR-like operator but decrease by 9.87 % and 3.20 % respectively with AND-like operator. For the test case "add4", "par", and "par2" we notice that hedges do not reduce the dead space area of the resulting floorplan when using the OR-like operator. The same thing is noticed in test cases "add4" and "par2" concerning both length and delay and with the same operator. Results obtained using the OR-like soft operator of the OWA approach are summarized in Table 7.10 which illustrates the effect of the parameter β on the individual objective criteria. The same hedges

Table 7.10: Results obtained using the OR-like soft operator with different values for β and hedges "more or less", "very", "very" applied to area, wirelength, and delay respectively.

Circuit	β	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
highway	0.0	1.36	3.37	2.07
	0.4	1.36	3.75	2.08
	0.6	1.28	3.66	2.10
	0.8	1.24	3.68	2.12
add4	0.0	1.35	3.21	1.94
	0.4	1.32	3.09	1.83
	0.6	1.42	3.33	1.91
	0.8	1.39	2.94	1.85
par	0.0	1.09	3.56	1.41
	0.4	1.18	3.69	1.43
	0.6	1.18	3.69	1.43
	0.8	1.04	3.79	1.44
par2	0.0	1.35	2.87	1.80
	0.4	1.21	3.13	1.86
	0.6	1.20	3.25	1.87
	0.8	1.12	3.16	1.84
chip	0.0	1.80	8.68	2.95
	0.4	1.92	8.99	3.06
	0.6	1.70	8.92	3.42
	0.8	1.68	10.02	3.42

as above are applied to the fuzzy rule again. As expected, larger values of β lead to floorplan solutions with smaller area in most of the test cases at the expense of a degradation with respect to the other two objectives (length and delay). In fact, this illustrates the flexibility of fuzzy logic in easily accommodating various design requirements and objectives. For example, in test case "par2" the dead space area decreases from 35 % to 12 % as the value of β increases from 0 to 0.8. In test case "add4", the dead space area fluctuates between 35 % and 39 % as β increases. The same for length and delay whose values do fluctuate as well.

Comparing the outcomes obtained with FGA and those obtained with WS-GA (see Table 7.4), we notice that the results are similar in most of the cases. For example, in test case "highway", the WS approach shows a floorplan with a dead space of 36 %. In contrast, the OR-like soft operator of the OWA approach in Table 7.6 leads to a floorplan with a dead space of 29%, i.e., a decrease of 19.44 %. However, for the same circuit, the normalized length increases by 2.47 %. The normalized delay is the same in both approaches. In test case "chip", the OR-like operator illustrates (see Table 7.6) an increase of the dead space to 85 % compared to the 71 % obtained with WS.

In fact, the quality of results using the OR-like soft operator without hedges in the rule is very close and in some cases better compared to the quality of results generated using WS. For the same circuit, and using the same operator but with hedges, as seen in Table 7.9, the dead space is 70 %. For the two other objectives, the normalized values are better than their counterpart of the WS-GA.

In conclusion, the fuzzification efforts are justified because the results obtained using fuzzy approach are colateral and sometimes better than those produced using the WS. Similar experiments are performed with two other well known iterative algorithms SA and TS. Finally, we will compare the outcomes of the three heuristics.

7.3 Results using Simulated Annealing

Simulated annealing (SA) was applied on the five circuits described in Table 7.2. The following perturbation schemes have been used:

1. Scheme 1: Inversion of a randomly selected chain of operators.
2. Scheme 2: Swapping of two randomly selected operands.
3. Scheme 3: Swapping of an operand and an operator.

Table 7.11 illustrates the results obtained using the simulated annealing optimization technique with the fuzzy rules modified by hedges "more or less", "very", and "very" applied to the linguistic values small area, short length , and low delay respectively. Different approaches have been used to build the cost function, i.e., Weighted Sum (WS-SA), probabilistic-like, and OR-like soft operator with β=0.6 and β=0.8. Compared to the WS-SA outcomes, the floorplan solutions of the Fuzzy Simulated Annealing (FSA) are within the same range. For example, for circuit "add4", the floorplan obtained with WS-SA shows 25 % of dead space area.

Table 7.11: Results of comparing WS-SA (Weighted Sum-Simulated Annealing), FSA-1 (*probabilistic – like*), FSA-2 (*additive combiner*), FSA-3 ($OR - like$ with β=0.6), and FSA-4 ($OR - like$ with β=0.8).

Circuit	Approach	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
highway	WS-SA	1.19	3.57	2.09
	FSA-1	1.30	2.98	1.79
	FSA-2	1.22	3.01	1.90
	FSA-3	1.23	3.46	1.80
	FSA-4	1.19	3.57	2.23
add4	WS-SA	1.25	2.23	1.61
	FSA-1	1.48	2.47	1.66
	FSA-2	1.10	2.64	1.73
	FSA-3	1.12	2.85	1.75
	FSA-4	1.07	3.47	2.16
par	WS-SA	1.01	3.35	1.38
	FSA-1	1.06	3.38	1.39
	FSA-2	1.00	3.45	1.40
	FSA-3	1.00	3.42	1.40
	FSA-4	1.00	3.62	1.42
par2	WS-SA	1.11	2.80	1.80
	FSA-1	1.17	2.56	1.77
	FSA-2	1.08	3.13	1.87
	FSA-3	1.09	3.02	1.83
	FSA-4	1.09	3.81	2.09
chip	WS-SA	1.46	7.71	2.75
	FSA-1	1.73	8.33	2.83
	FSA-2	1.54	7.66	2.97
	FSA-3	1.57	8.42	3.14
	FSA-4	1.59	9.27	3.31

However, with FSA-2-3-4 the amount of dead space is of the order of 10 %, 12 %, and 7 % respectively. For test case "chip", the WS-SA performs slightly better compared to FSA. The obtained floorplan has 46 % of dead space. The normalized values for length and delay obtained using FSA-2 show similar quality with respect to those obtained using WS. In test case "par2", WS-SA leads to a floorplan with 11 % of dead space. FSA using different approaches generates better results, i.e., floorplans having 8% and 9% of dead space. The normalized values for the two other objectives show a better quality with the WS-SA compared to the FSA. Figure 7.7 shows the optimal floorplan obtained for circuit "highway" when using probabilistic-like operators and hedges in rule R3 from where the expression giving the cost is derived. Figures 7.8 and 7.9 depict the variations in the values of the best cost and the cost respectively obtained for circuit "highway". Figures 7.10, 7.11, and 7.12 show the variations in the membership values of area, length, and delay respectively obtained for the same circuit.

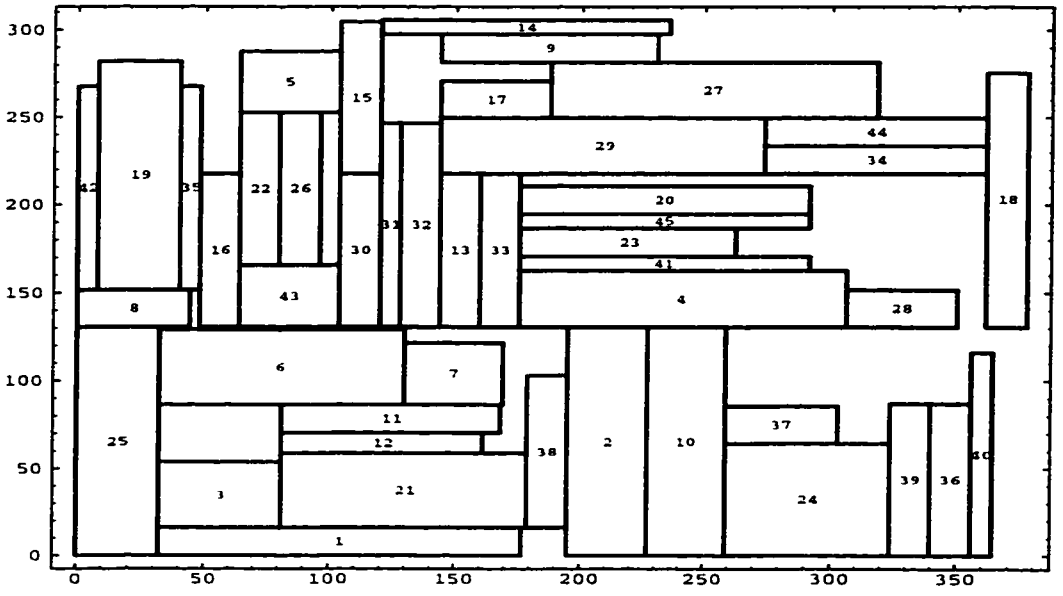


Figure 7.7: Best Floorplan solution of circuit "highway" obtained using FSA with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

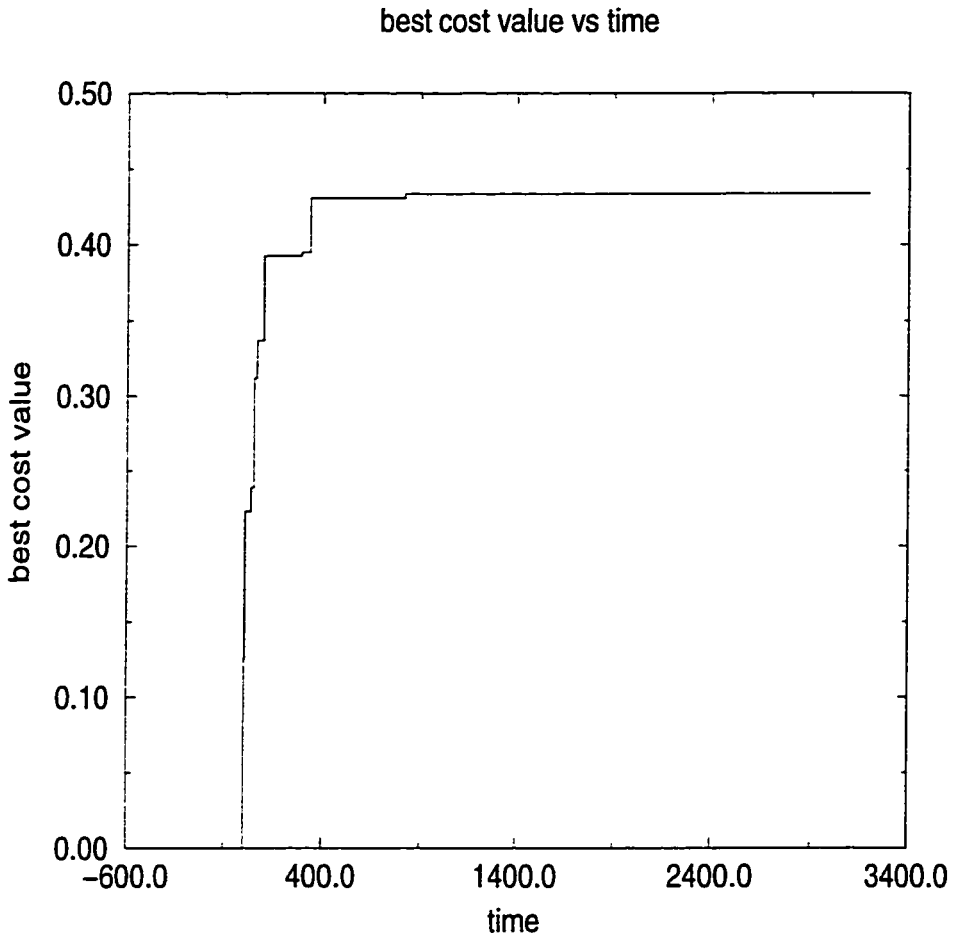


Figure 7.8: Best cost value vs time obtained using FSA with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively..

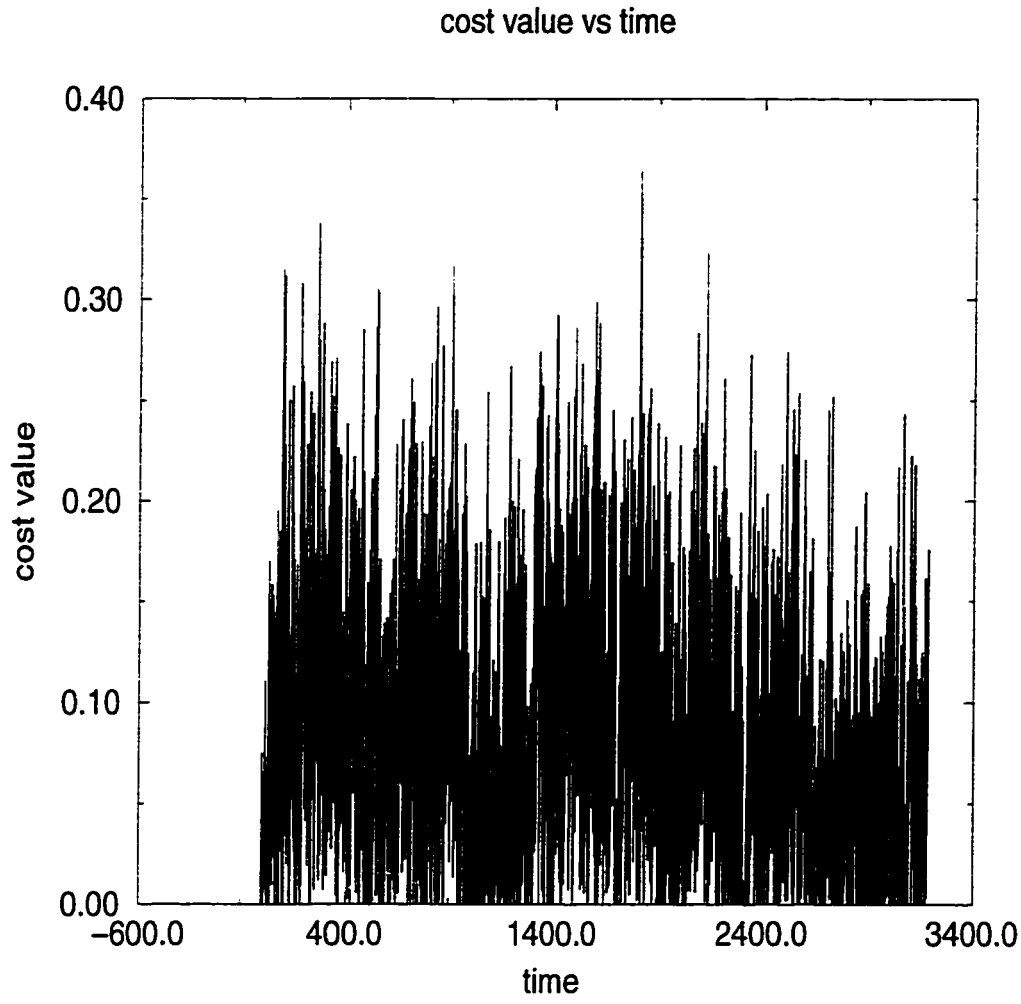


Figure 7.9: Cost value vs time obtained using FSA with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

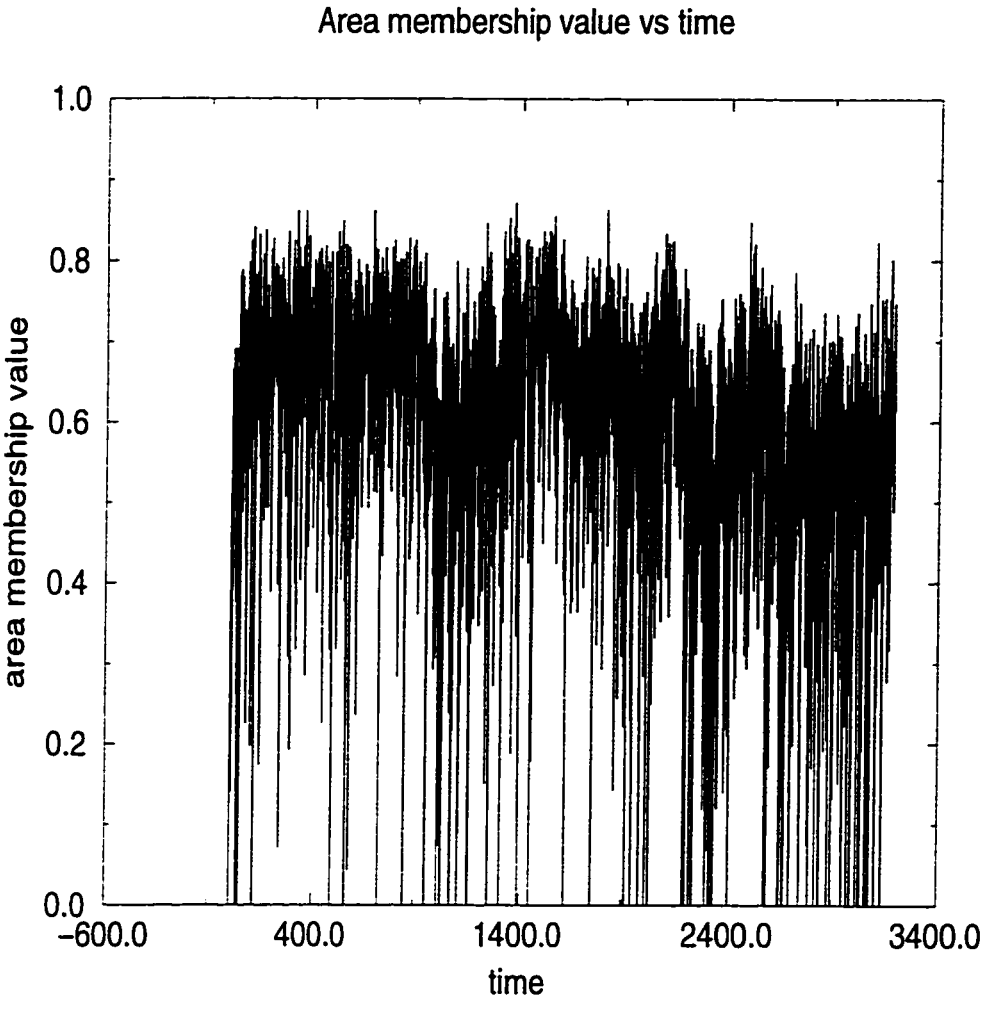


Figure 7.10: Area membership value vs time obtained using FSA with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

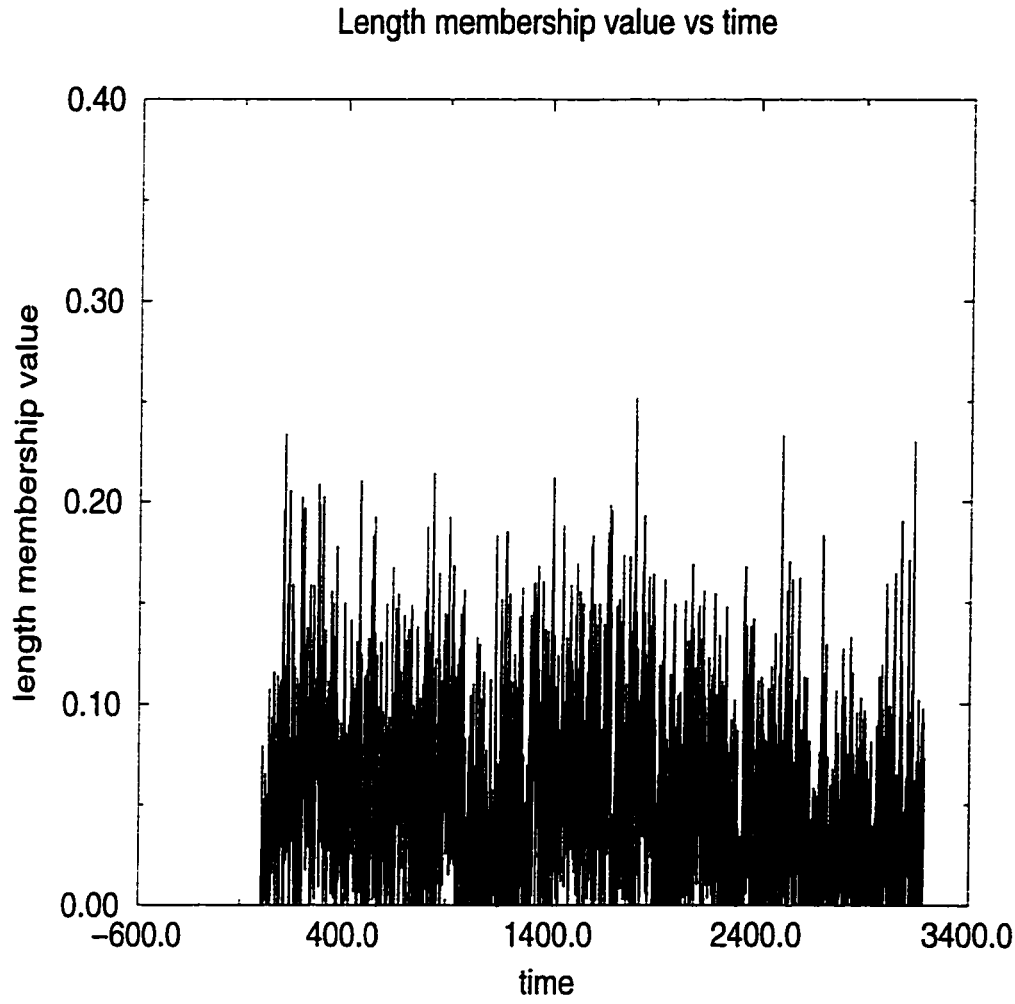


Figure 7.11: Length membership value vs time obtained using FSA with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

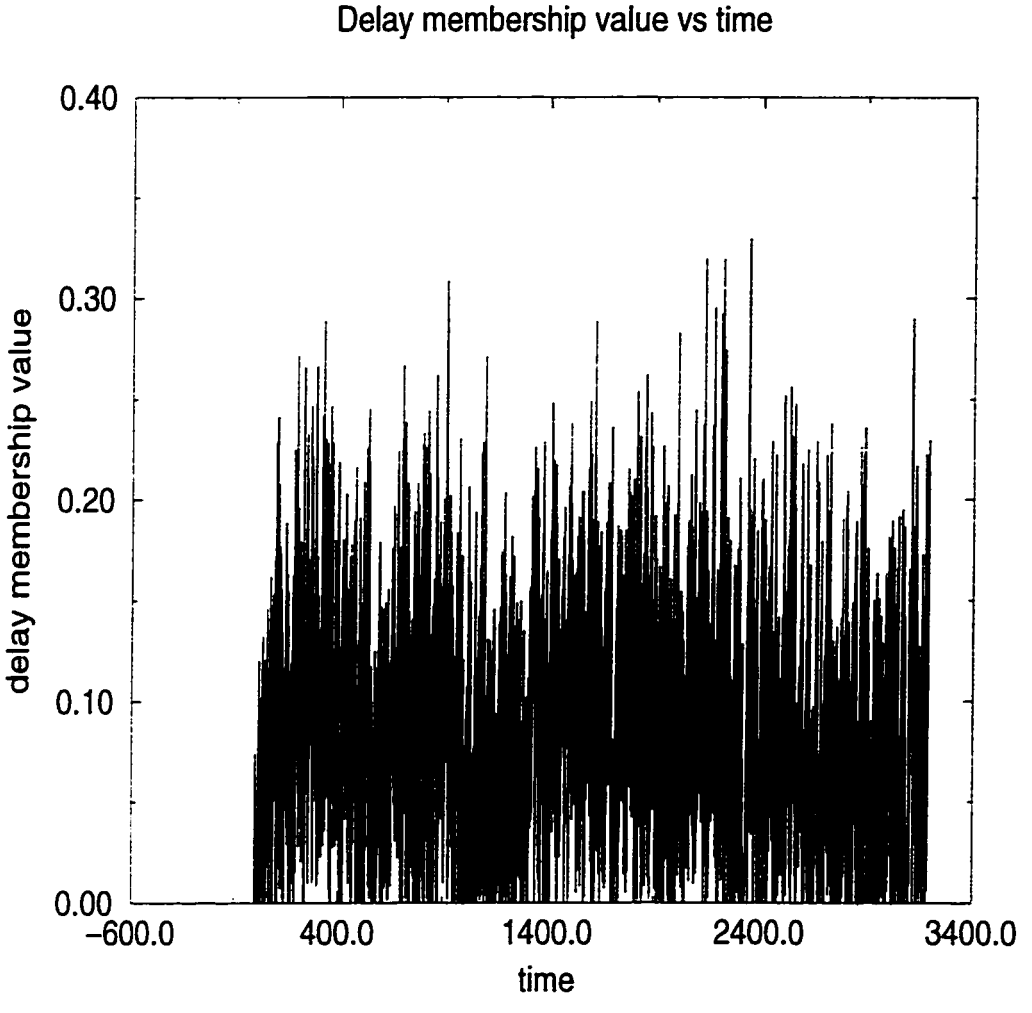


Figure 7.12: Delay membership value vs time obtained using FSA with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

7.4 Results using Tabu Search

Tabu Search (TS) algorithm was applied on the five circuits described earlier. Results are summarized in Table 7.2. The programs were run for 1000 iterations. In each iteration, 20 neighbor solutions were generated. The best cost corresponding to one of these generated solutions, within each iteration, was considered. The perturbation schemes to generate neighbor solutions, are the same as those used with the simulated annealing approach.

1. Scheme 1: Inversion of a randomly selected chain of operators.

2. Scheme 2: Swapping of two randomly selected operands.

3. Scheme 3: Swapping of an operand and an operator.

The size of the tabu list depends on the test case. For each circuit, the tabu list size was set to $\frac{N}{4}$ where N is the number of basic blocks.

For each perturbation scheme, a separate tabu list is associated. In our case, three tabu lists T1, T2, and T3 are defined for scheme 1, scheme 2, and scheme 3 respectively. The attributes stored in the tabu lists are related to each perturbation technique. In scheme 1, where the perturbation consists of inverting a chain of operators, the attribute consists of the position of the fist operator of the chain to be stored in the next free slot of T1. For scheme 2, the attribute consists of the left operand involved in the swap to be stored in T2. The attribute related to scheme 3 consists of the operand, which is swapped with an operator, to be stored

in T3. For each test case, the three tabu lists T1, T2, and T3 have the same size.

Table 7.12: Results of comparing WS-TS (Weighted Sum-Tabu Search), FTS-1 (*probabilistic − like*), FTS-2 (*additive combiner*), FTS-3 ($OR - like$ with β=0.6), and FTS-4 ($OR - like$ with β=0.8).

Circuit	Approach	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
highway	WS-TS	1.10	3.16	1.89
	FTS-1	1.22	2.85	1.77
	FTS-2	1.23	2.81	1.66
	FTS-3	1.15	3.15	1.73
	FTS-4	1.13	3.03	1.70
add4	WS-TS	1.15	2.52	1.52
	FTS-1	1.22	2.14	1.55
	FTS-2	1.09	2.39	1.51
	FTS-3	1.15	2.67	1.58
	FTS-4	1.26	2.14	1.60
par	WS-TS	1.02	3.63	1.42
	FTS-1	1.04	3.47	1.40
	FTS-2	1.03	3.58	1.41
	FTS-3	1.03	3.65	1.42
	FTS-4	1.02	3.60	1.42
par2	WS-TS	1.10	2.91	1.83
	FTS-1	1.40	2.16	1.65
	FTS-2	1.10	2.87	1.80
	FTS-3	1.10	2.88	1.80
	FTS-4	1.10	2.90	1.80
chip	WS-TS	1.49	6.02	2.23
	FTS-1	1.35	5.78	1.89
	FTS-2	1.33	5.80	2.13
	FTS-3	1.36	6.15	2.37
	FTS-4	1.22	5.66	2.17

The results obtained using TS technique are summarized in Table 7.12 with the fuzzy rules modified by hedges "more or less", "very", and "very" applied to the linguistic values small area, short length , and low delay respectively. Different approaches have been used to build the cost function, i.e., Weighted Sum (WS-TS), probabilistic-like, and OR-like soft operator with β=0.6 and β=0.8. The best floorplan, from the area point of view, obtained by FTS-4 approach for test case "highway" has an area equal to 1.13 that of the smallest possible area, i.e., a dead space of 13 %. However, the best length and delay values obtained are with FTS-2 for the same circuit and are 2.81 and 1.66 that of the smallest possible length and delay respectively. In the two cases, there was an increase of 8.85 % of floorplan area and a decrease of the order of 7.26 % and 2.35 % for the length and delay respectively. We notice that the amount of increase of the area obtained by FTS-2 is approximately the same as the amount of decrease of the length using FTS-4 approach. Another example with test case "par2" shows that FTS-2, FTS-3, and FTS-4 lead to the same results. On the other hand, the outcome due to FTS-1 shows an area having 40 % of dead space, i.e., an increase of the order of 27.27 %. This increase of area is compensated by a decrease of both length and delay of the order of 25 % and 8.33 % respectively. Hence, the amount of increase of the area is compensated by the amount of decrease of both length and delay. Comparing the results obtained using fuzzy logic approaches and those obtained by the weighted sum approach we notice that they are the same. As illustration of the results, the following figures show what we obtained for the test case "highway". Figure 7.13

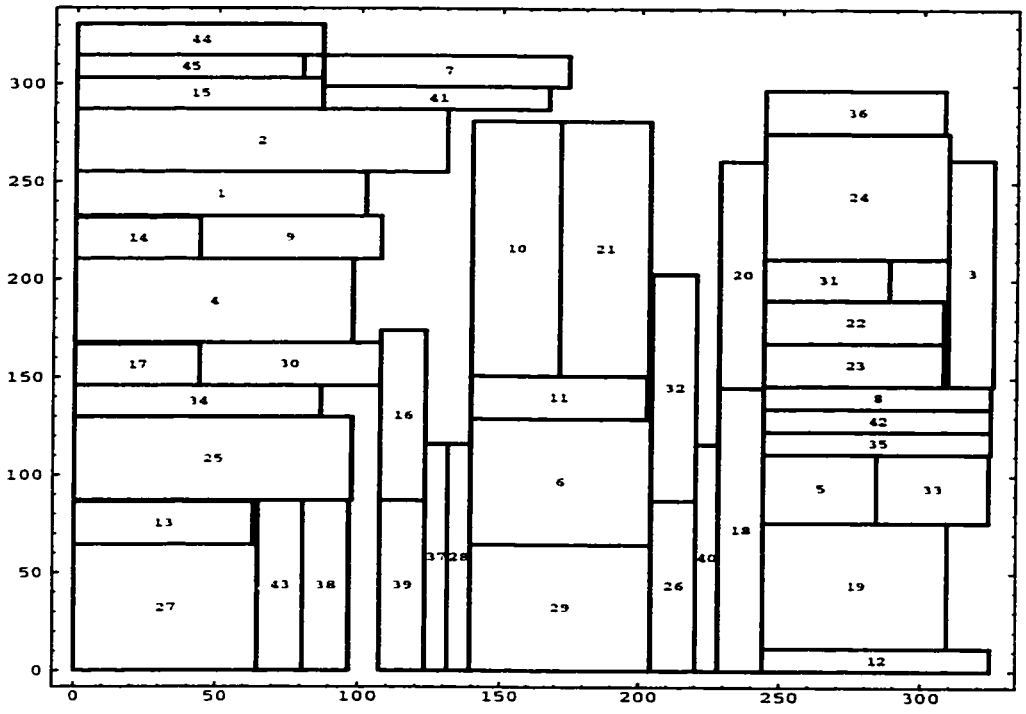


Figure 7.13: Best Floorplan solution obtained using FTS with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

shows the optimized floorplan layout which is our solution obtained using FTS with the probabilistic-like approach. In Figures 7.14, 7.15, 7.16, the variation of the membership values of area, length, and delay respectively are plotted against time during which the optimization process was carried out. The Best cost in each iteration, provided that the corresponding move is accepted, is always increasing. This is shown in Figure 7.17 where a non-decreasing curve representing the best cost is plotted versus time.

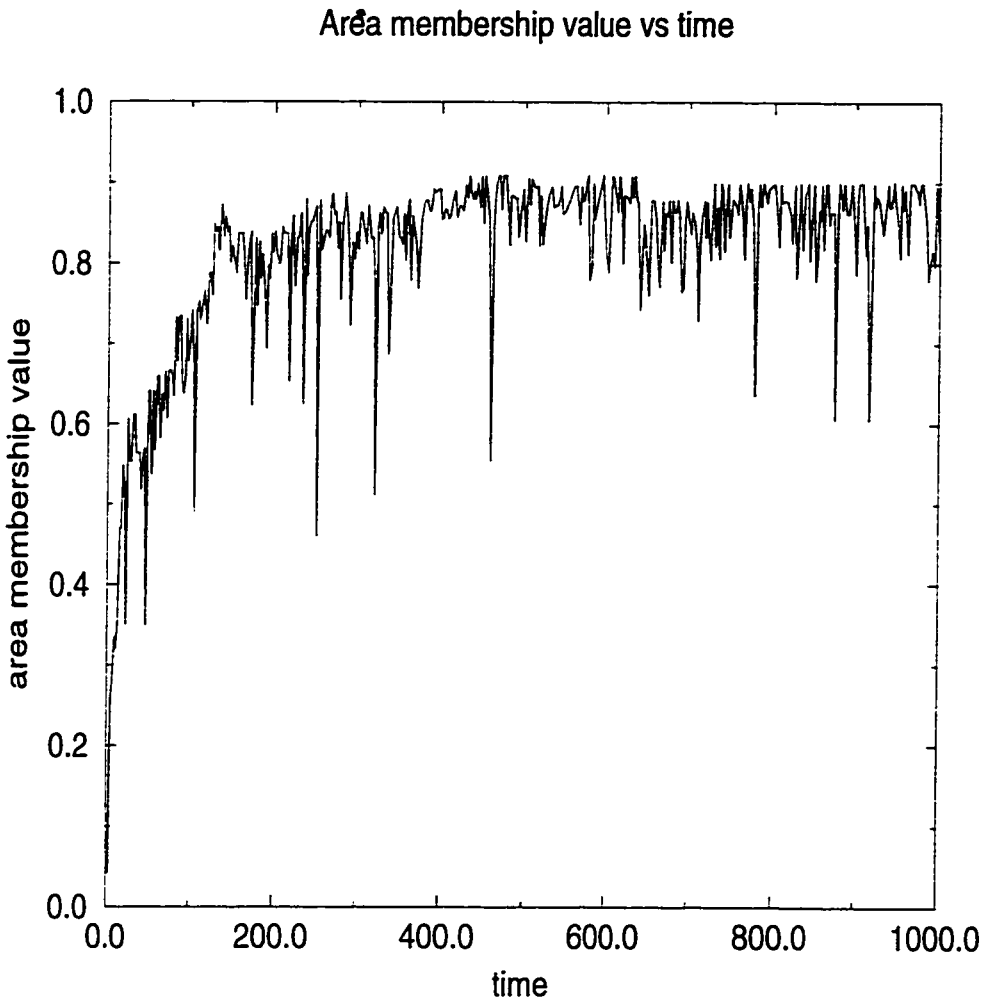


Figure 7.14: Area membership value vs time obtained using FTS with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

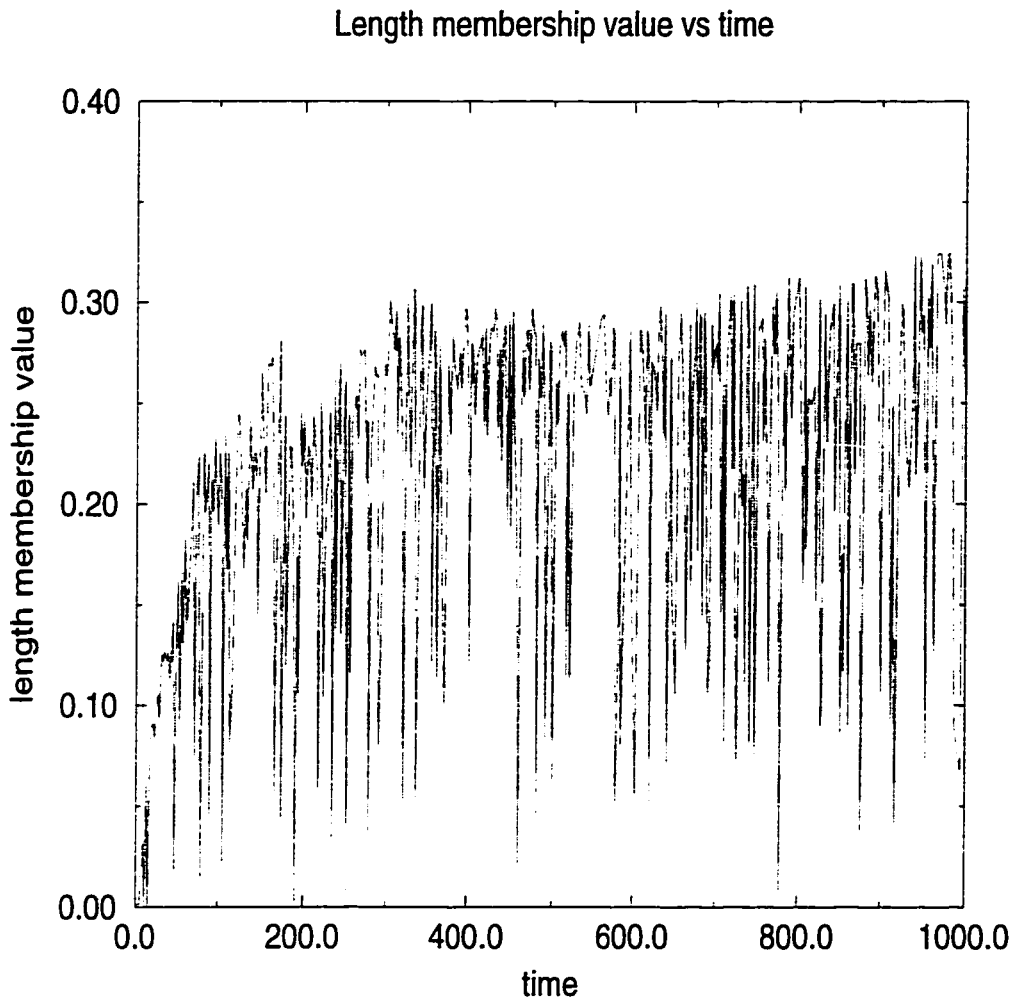


Figure 7.15: Length membership value vs time obtained using FTS with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

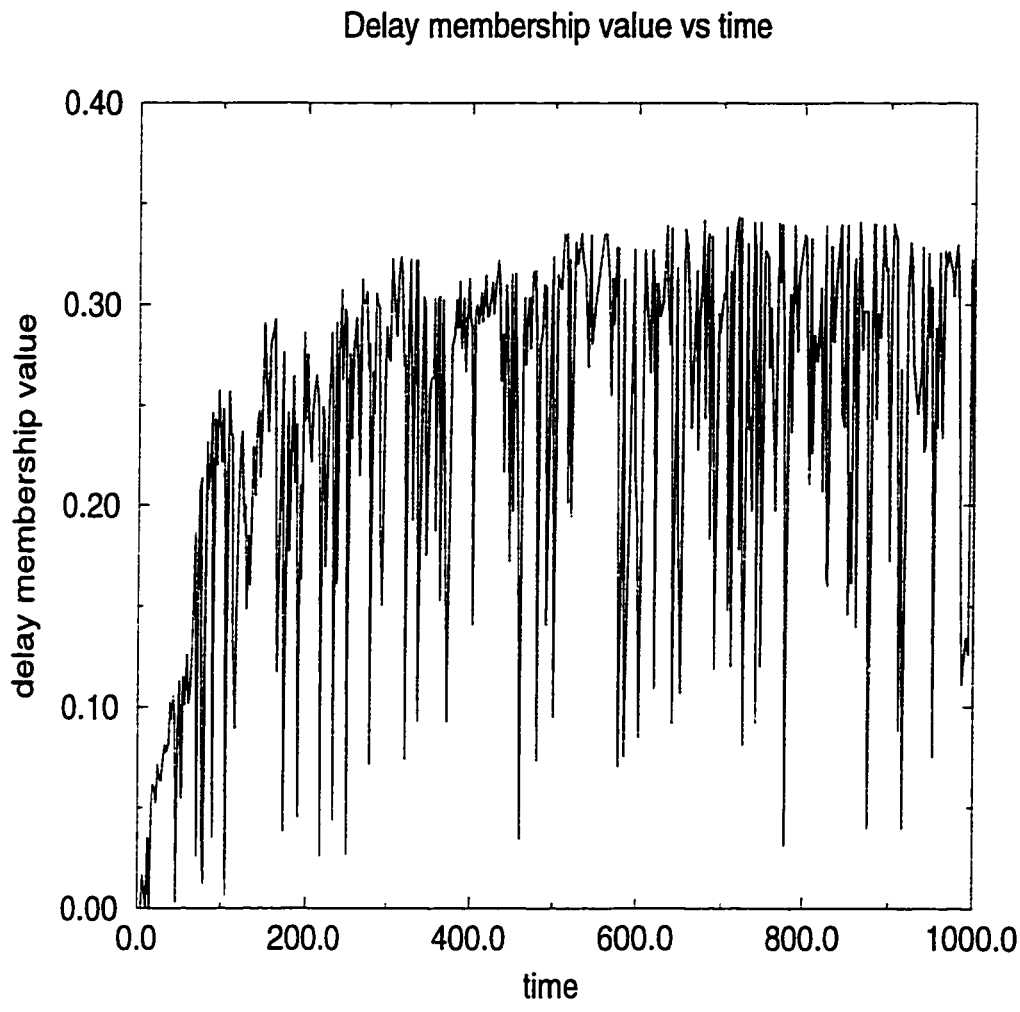


Figure 7.16: Delay membership value vs time obtained using FTS with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

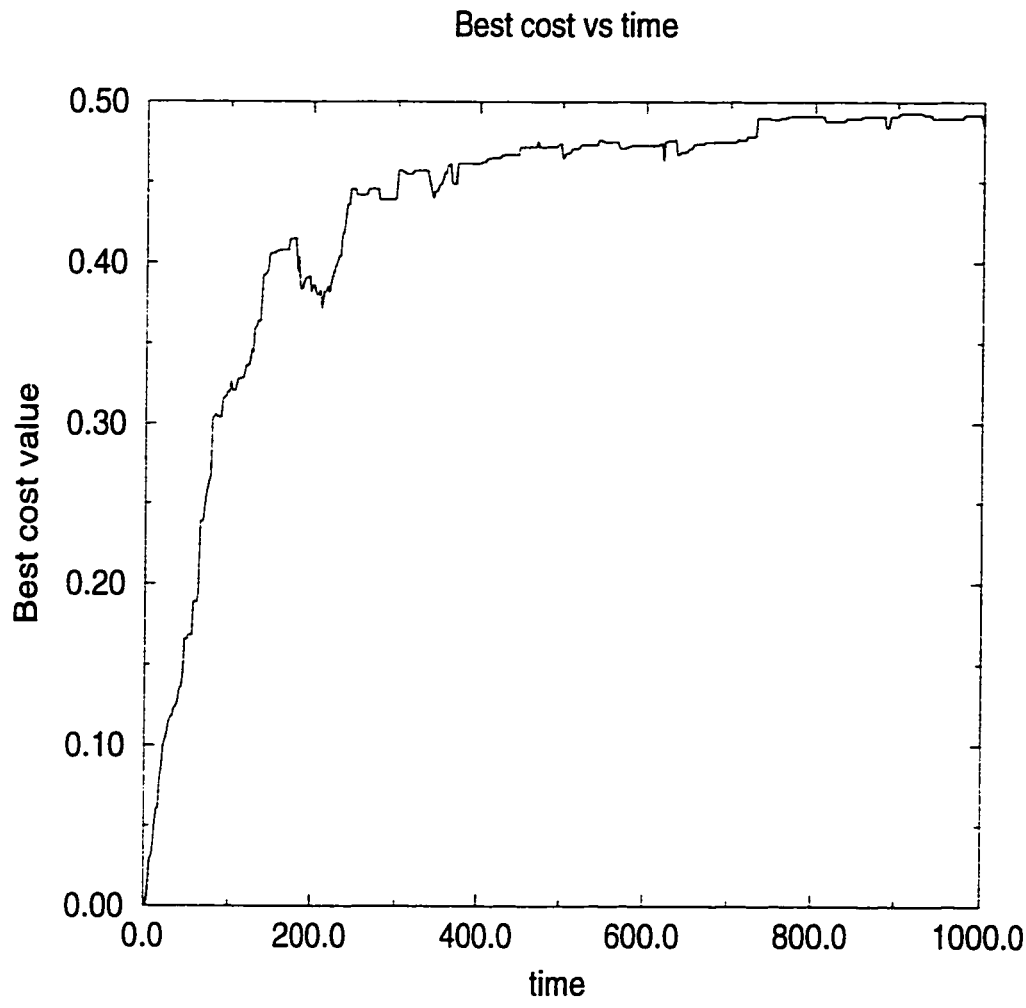


Figure 7.17: Best cost value vs time obtained using FTS with the probabilistic-like operators and hedges "more or less" "very" and "very" applied to area, length, and delay respectively.

7.5 Comparison of Results Obtained with FGA, FSA, and FTS

In Table 7.13 and Table 7.14, the floorplan solutions obtained using different fuzzy approaches and operators are illustrated. In Table 7.13, we notice that in most test cases, the FTS outcomes are better in quality than those produced by FGA and FSA. For example, with the probabilistic-like approach applied to circuit "chip", FTS generates an area with a dead space of 35 %, a normalized length of 5.78 and a normalized delay of 1.89. Comparing these results with both FGA and FSA outcomes for the same circuit, we record an improvement of the order of 36.62 % and 21.96 % for the area, 37.85 % and 30.61 % for the length, and 42.55 % and 33.21 % for the delay compared to FGA and FSA results respectively.

In some cases such as when using the additive combiner approach, for circuit "par", the FSA provides better results than FTS, but both are still comparable in term of quality. The same thing can be observed concerning circuit "par2". FSA generates a better area with respect to FTS, however, FTS generates better length and delay. Table 7.14 shows floorplan solutions generated using the OWA approach. We notice that FSA and FTS outcomes are better than those obtained using FGA. As an example, when β=0.6 for circuit "highway", FTS generates a dead space area of 15 % compared to the 28 % and 23 % produced by FGA and FSA respectively. The same thing can be observed concerning the length with an improvement of 13.93 % and 8.93 % and the delay with an improvement of

Table 7.13: Results obtained with (*probabilistic – like*) and (*additive combiner*) approaches using FGA, FSA, and FTS

Circuit	Approach	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
probabilistic-like	FGA (highway)	1.42	3.75	2.12
	FSA (highway)	1.30	2.98	1.79
	FTS (highway)	1.22	2.85	1.77
	FGA (add4)	1.34	3.05	1.76
	FSA (add4)	1.48	2.47	1.66
	FTS (add4)	1.22	2.14	1.55
	FGA (par)	1.23	3.52	1.40
	FSA (par)	1.06	3.38	1.39
	FTS (par)	1.04	3.47	1.40
	FGA (par2)	1.42	2.87	1.78
	FSA (par2)	1.17	2.56	1.77
	FTS (par2)	1.40	2.16	1.65
	FGA (chip)	2.13	9.30	3.29
	FSA (chip)	1.73	8.33	2.83
	FTS (chip)	1.35	5.78	1.89
additive combiner	FGA (highway)	1.36	3.71	2.08
	FSA (highway)	1.22	3.01	1.90
	FTS (highway)	1.23	2.81	1.66
	FGA (add4)	1.32	3.10	1.83
	FSA (add4)	1.10	2.64	1.73
	FTS (add4)	1.09	2.39	1.51
	FGA (par)	1.18	3.68	1.43
	FSA (par)	1.00	3.45	1.40
	FTS (par)	1.03	3.58	1.41
	FGA (par2)	1.21	3.13	1.86
	FSA (par2)	1.08	3.13	1.87
	FTS (par2)	1.10	2.87	1.80
	FGA (chip)	1.92	8.98	3.06
	FSA (chip)	1.54	7.66	2.97
	FTS (chip)	1.33	5.80	2.13

17.62 % and 3.88 % using FGA and FSA respectively. Comparing the results obtained with β=0.6 and β=0.8, Table 7.14 shows that as β increases, the dead space decreases in most of the test cases. However, the change is unpredictable for both length and delay. Overall FTS generated better floorplan than FGA and FSA.

Table 7.14: Results obtained with ($OR-like$ with β=0.6), and ($OR-like$ with β=0.8) approaches using FGA, FSA, and FTS.

Circuit	Approach	$\frac{Area}{Area_{min}}$	$\frac{Length}{Length_{min}}$	$\frac{Delay}{Delay_{min}}$
OR-like-soft β=0.6	FGA (highway)	1.28	3.66	2.10
	FSA (highway)	1.23	3.46	1.80
	FTS (highway)	1.15	3.15	1.73
	FGA (add4)	1.42	3.33	1.91
	FSA (add4)	1.12	2.85	1.75
	FTS (add4)	1.15	2.67	1.58
	FGA (par)	1.18	3.69	1.43
	FSA (par)	1.00	3.42	1.40
	FTS (par)	1.03	3.65	1.42
	FGA (par2)	1.20	3.25	1.87
	FSA (par2)	1.09	3.02	1.83
	FTS (par2)	1.10	2.88	1.80
	FGA (chip)	1.70	8.92	3.42
	FSA (chip)	1.57	8.42	3.14
	FTS (chip)	1.36	6.15	2.37
OR-like-soft β=0.8	FGA (highway)	1.24	3.68	2.12
	FSA (highway)	1.19	3.57	2.23
	FTS (highway)	1.13	3.03	1.70
	FGA (add4)	1.39	2.94	1.85
	FSA (add4)	1.07	3.47	2.16
	FTS (add4)	1.26	2.14	1.60
	FGA (par)	1.04	3.79	1.44
	FSA (par)	1.00	3.62	1.42
	FTS (par)	1.02	3.60	1.42
	FGA (par2)	1.12	3.96	1.84
	FSA (par2)	1.09	3.81	2.09
	FTS (par2)	1.10	2.90	1.80
	FGA (chip)	1.68	10.02	3.42
	FSA (chip)	1.59	9.27	3.31
	FTS (chip)	1.22	5.66	2.17

7.6 Comparison of GA, SA, and TS

The GA is more complex than both SA and TS from the programming point of view. The different operators such as crossovers, mutations, and inversion as well as the selection of the next generation population require more coding to allow a correct implementation of the algorithm. Both SA and TS can be implemented by easily translating their algorithms into the equivalent program code. The three programs (GA, SA, and TS) use the same memory space to solve the problem due to the fact that one floorplan is processed at a time.

In our implementation of GA and TS, there is a fixed number of iterations and hence the overall time to solve the problem depends on the number of these iterations. If the number of iterations in both programs are set to the same value, then the overall time to solve the problem would have been approximately the same as well (this is true in the case where the number of neighbor solutions generated in TS is approximately equal to the number of parents and the number of offsprings added together). In SA, the programs stop if the stopping criteria are met. In many cases, these stopping criteria were not met even after days of program run. The results were collected when either no noticeable improvement was recorded or when the quality of results becomes better or almost the same as the results produced by the SA-WS approach.

The GA is hard to tune since good values for the probabilities of mutation, crossover, and inversion have to be determined. In SA, we need to select a proper

value for r (cooling rate) and to determine the initial temperature using a reasonable value or generating it with a heuristic. In TS, we need to tune the program by selecting an acceptable size for the tabu lists such that the results are satisfactory. The tuning tasks require some efforts in terms of preleminary trials and experiments.

7.7 Conclusion

In this chapter, we described the implementation of cost functions using fuzzy rules. For each linguistic value of a linguistic variable, a membership function was defined. Different fuzzy operators have been used. Modifiers such as hedges and preference rules were applied in the appropriate context. The cost functions derived from fuzzy rules were applied to GA, SA, and TS. This resulted into FGA, FSA, and FTS. Finally, experimental results were presented and discussed.

Floorplan solutions were computed using the Weighted Sum (WS) approach for each type of heuristic. The purpose was to compare the outcomes of this approach to generate a cost function against the results obtained through the fuzzified heuristics FGA, FSA, and FTS. In each case, different operators have been tested in order to derive an appropriate cost function from the fuzzy rules and see which operator produces best results. As illustrated in the different tables, we notice that WS approach and Fuzzy Logic approach generate comparable results which are sometimes very close from the quality point of view. However, it is important to

mention that the quality of the outcomes obtained through fuzzy heuristics is very sensitive to the type of fuzzy operator applied to generate the cost function. According to the previous tabulated results, we can deduce that both compensatory and non-compensatory operators produce satisfying results. However, the quality of these results do not justify the use of Fuzzy Logic to solve such a problem when compared with the quality of the outputs of the WS approach. On the other hand, the other approaches such as probabilistic-like, additive combiner, and OWA come up with results that suggest the fact that the use of Fuzzy Logic is really a successful entreprise. In many test cases, the results produced by the fuzzified heuristics outperform those produced using WS technique. In FSA and FTS, the results were of similar quality and sometimes better than those produced by the WS method as listed in the corresponding tables. The comparison between the outcomes generated by the three fuzzy heuristics shows similar quality of results between FSA and FTS. These results, however, are slightly better than those produced by FGA. The reason comes from the fact that GA is more difficult to tune than SA or TS. Also, the GA outcomes are sensitive to the strategy of selection of the individuals of the next generation. Exploring the different possibilities and strategies can cause the GA results to improve in their quality. Also, the number of generations can vary from one circuit to another depending on its size. Increasing the number of generations, in some cases, may lead to better results.

Chapter 8

Conclusion

8.1 Summary

The use of Fuzzy Logic is expected to remove any ambiguity in a multi-objective optimization problem in the sense that it can express the relation between different objectives as a linear combination with fuzzy operators. Floorplan solutions produced by weighted sum cost functions and cost functions derived from fuzzy rules were of similar quality with respect to all three objective criteria. From our experience in this work, we can conclude the following about the different aspects that were of concern to us.

1. In FGA, FSA, and FTS the selection of the fuzzy rules were straightforward. The emphasis on appropriate objective criteria is embodied within the fuzzy operators and/or fuzzy preference rules. The quality of solution was sensitive mostly to the type of fuzzy operators rather than the rules used or the shape

of membership functions, as long as the rules express what the designer really desires and the membership functions are nonincreasing.

2. In FGA, FSA, and FTS, the designer's preferences are easily and naturally described in linguistic terms. The resulting cost equation is a direct consequence of the application of the defined fuzzy rules and fuzzy algebra. The arbitrariness associated with the selection of weights is removed.

3. As the number of objectives increases and designer requirements get more complex, it becomes less and less obvious how such aspects would get accommodated using a simple weighted sum of the various objectives. In contrast, fuzzy algebra easily accommodates any design goals and any number of objective criteria.

4. The weighted sum approach as implemented in this work is a special form of addition.

5. Since the weighted sum approach is very sensitive to the weight values, it is not clear how one would make the algorithm adaptive as the search progresses. However, such adaptive aspects can easily be accommodated in the fuzzy approach.

The ultimate criteria for effectiveness of fuzzy logic in design is practice, i.e., whether or not quality of solutions obtained by fuzzy logic tools is higher than solutions generated by tools that do not use fuzzy logic. Also, the gain certainly

should be weighed against efforts required for the incorporation of fuzzy logic into design tools, their flexibility to adaptation and user friendliness. Based on our experience, such efforts are justifiable.

8.2 Future Work

As a future work, multiple fuzzy rules can be used instead of a single one. This necessitates the building of an engine that will initiate the fuzzy inference. When the inputs are given, the output is calculated by means of fuzzy inference. The output is the value of the cost that will guide the search. Another possible work consists of developping a fuzzy algorithm as an ordered set of fuzzy instructions which upon execution yield an approximate solution to our floorplanning problem. Finally, it is possible to extend this floorplanning problem to nonslicing structures and solve it using fuzzy approach.

Bibliography

[1] C. Alsina, *As You Like Them: Connectives in Fuzzy Logic*, universitat oberta de catalogna.

[2] T. Chao and Y. Hsu, *Rectilinear Steiner Tree Construction By Local and Global Refinement*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, No. 3, pp. 303-309, March 1994.

[3] C. Chen and I. Tollis, *Area Optimization of Spiral Floorplans*, J.Circuits, Syst., Comput., Vol. 3, No. 4, pp. 833-857, 1993.

[4] Chen et al, *Efficient Approximation Algorithms for Floorplan Area Minimization*, Proceedings Design Automation Conference 1996.

[5] K. Chong and S. Sahni, *Optimal Realization of Floorplans*, IEEE Trans. Computer-Aided Design, Vol. 12, pp. 793-801, June 1993.

[6] J.P.Cohoon et al, *Distributed Genetic Algorithm for the Floorplanning Design Problem*, IEEE Transactions on Computer-Aided Design, Vol. 10, No. 4, pp. 483-492, April 1991.

[7] E. Cox, *Fuzzy Fundamentals*, IEEE Spectrum, pp. 58-61, 1992.

[8] E. Czogala, *Probabilistic Sets in Decision Making and Control*, Koln:Verlag TUV Rheinland, 1984.

[9] W. Dai, *Hierarchical Placement and Floorplanning in BEAR*, IEEE Transactions on Computer-Aided Design. Vol. 8. No. 12, pp. 1335-1349, December 1989.

[10] D. Dubois and H. Prade, *Fuzzy Sets and Systems Theory and Applications*, by Academic Press, Inc, 1980.

[11] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, by Addison-Wesley Publishing Company, INC, 1989.

[12] M. Jamshidi, *On Software and Hardware Applications of Fuzzy Logic*, p 376, in *Fuzzy Sets, Neural Networks and Soft Computing*, Edited by: R. R. Yager and L. A. Zadeh, by Van Nostrand Reinhold, 1994.

[13] R. Jayabharathi and M. Manjoul, it Fuzzy Logic for Standard Cell Placement, Cybernetics and Systems: An International Journal, 24: 197-215, 1993.

[14] E. Kang, R. Lin, and E. Sharagowitz , *Fuzzy Logic Approach to VLSI Placement*, IEEE Transactions On VLSI, Dec 1994, Vol. 2, No. 4, pp. 489-501

[15] C. Karr, *Adaptive Control with Fuzzy Logic and Genetic Algorithms*, p. 345, in *Fuzzy Sets, Neural Networks and Soft Computing*, Edited by: R. R. Yager and L. A. Zadeh, by Van Nostrand Reinhold, 1994.

[16] S. Kirkpatrick et al. *Optimization by Simulated Annealing*, Science, 220 pp. 498-516, May 1983.

[17] S. Kung, *Digital Neural Networks*, 1993 by PTR Prentice-Hall, Inc.

[18] J. Mendel, *Fuzzy Logic Systems for Engineering: A Tutorial*, Proceedings of the IEEE, Vol. 83, No. 3, March 1995.

[19] T. Murata and H. Ishibushi, *MOGA Multiobjective Genetic Algorithms*, Proceedings of International Conference on Evolutionary Computation, pp. 289-294, 1995.

[20] Foresight Manual, Orbit Semiconductor Inc, Sunnyvale California, July 1992.

[21] R. Otten, and L. Van Ginneken, *Floorplan Design using Simulated Annealing*, Proc. ICCAD 84, (1984), pp. 96-98.

[22] N. Pal and J. Rezkek, *Measures of Fuzziness: A Review and Several New Classes*, in *Fuzzy Sets, Neural Networks and Soft Computing*, Edited by: R. R. Yager and L. A. Zadeh, by Van Nostrand Reinhold, 1994.

[23] P. Pan, and C. Lui, *Area Minimization for Floorplans*, IEEE Trans. on Computer-Aided Design, Vol. 14, pp. 123-32, Jan 1995.

[24] W. Pedrycz et al. *A Fuzzy Cognitive Structure: Foundations, Applications, and VLSI Implementation*, in *Fuzzy, Holographic, and Parallel Intelligence* , Edited by Branko Soucek and the IRIS Group, by John Wiley and Sons, Inc. 1992.

[25] M. Rebaudengo and M. Reorda, *GALLO: A Genetic Algorithm for Floorplanning Area Optimization*, IEEE Trans. on Computer-Aided Design, Vol. 15, No. 8, pp. 943-951, Aug 1996.

[26] J. Ribiero, P. Treleaven, and C. Alippi, Genetic Algorithm Programming Environments, Computer, pp. 28-35, 1994.

[27] Sadiq M. Sait and Habib Youssef, *VLSI Physical Design Automation: Theory and Practice*, Mc Graw-Hill International (UK) Limited, 1995.

[28] Sadiq M. Sait, Habib Youssef, Tanvir Shahid, Benten. M. S. T. *Timing Influenced General-Cell Genetic Floorplanner*, Proc 1995 IEEE Asia South Pac Des Autom Conf ASP DAC 95, 1995, pp. 135-140.

[29] Sadiq M. Sait and Habib Youssef, *Timing Driven Genetic Floorplanner*, to appear in the International Journal on Microelectronics.

[30] Sadiq M. Sait and Habib Youssef, *Iterative Computer Algorithms and Their Applications in Engineering*, IEEE CS Press, to appear, 1998.

[31] J. Schaffer, *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*, Proceedings of International Conference on GAs, pp. 93-100, 1985.

[32] K. Shahookar and P. Mazumder, *VLSI Cell Placement*, ACM Computing Surveys, Vol. 23, No. 2, 1991

[33] W. Shi, *A Fast Algorithm for Area Minimization of Slicing Floorplans*, IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 15, No. 12, pp. 1525-1532, December 1996.

[34] M. Srinivas and L. Patnaik, *Genetic Algorithms: A Survey*, Computer, pp. 17-26, 1994.

[35] L. Stockmeyer, *Optimal Orientation of Cells in Slicing Floorplan Designs*, Information and Control, Vol. 57, pp. 91-101, 1983.

[36] S. Sutanthavibul et al, *An Analytical Approach to Floorplan Design and Optimization*, IEEE Transactions on Computer-Aided Design, Vol. 10, No. 6, pp. 761-769, June 1991.

[37] T. Terno, K. Asai, and M. Sugeno, *Fuzzy Systems Theory and Its Applications*, Academic Press, Inc, 1992.

[38] T. Wang and D. Wong, *Optimal Floorplan Area Optimization*, IEEE Trans. on Computer-Aided Design, Vol. 11, pp. 992-1002, Aug 1992.

[39] T. Wang, and D. Wong, *Graph-based techniques to speed up floorplan area optimization*, Integration, the VLSI journal 15 (1993), pp. 179-199.

[40] Wang et al, *Floorplan Area Optimization Using Network Analogous Approach*, Proceedings-IEEE International Symposium On Circuits and Systems, Vol. 1, 1995.

[41] S. Wimer, I. Koren, and I. Cederbaum, *Optimal Aspect Ratios of Building Blocks in VLSI*, IEEE Trans. on Computer-Aided Design, Vol. 8, pp. 139-145, 1989.

[42] D. Wong, H. Leong, and C. Liu, *Simulated Annealing for VLSI Design*, Boston, MA, Kluwer Academic, 1988.

[43] D. Wong and C. Liu, *A New Algorithm For Floorplan Design*, 23rd Design Automation Conference, IEEE (1986), pp. 101-105.

[44] R. Yager, *Fuzzy Sets and Approximate Reasoning in Decision and Control*, 1992 IEEE.

[45] R. Yager, *Ordered Weighted Averaging (OWA)*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, pp. 183-190, 1988.

[46] G. Yeap and M. Sarrafzadeh, *A Unified Approach To Floorplan Sizing And Enumeration*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 12, No. 12, pp. 1858-1867, December 1993.

[47] L. Zadeh, *Outline of a New Approach to the analysis of Complex Systems and Decision Processes*, IEEE Transactions On Systems, Man, and Cybernetics, Vol. SMC-3, No. 1, Jan 1973.

[48] L. Zadeh, *Fuzzy Sets*, Information and Control. Vol. 8, pp. 338-353 (1956)

[49] L. Zadeh, *The Concept of a Linguistic Variable and its Application to Approximate Reasoning*, Information Sciences Vol. 8, pp. 199-249 (1975).

[50] L. Zadeh, it Fuzzy Logic = Computing with Words, IEEE Transactions On Fuzzy Systems, Vol. 4, No. 2, pp. 103-111, May 1996.

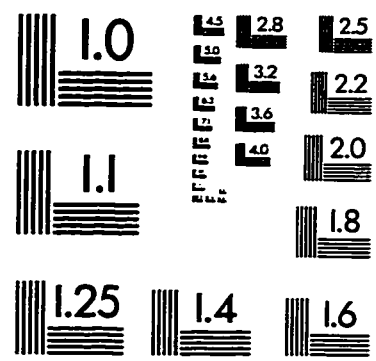
[51] L. Zadeh et al, Fuzzy Sets And Their Applications To Cognitive And Decision Processes, 1975, By Academic Press.

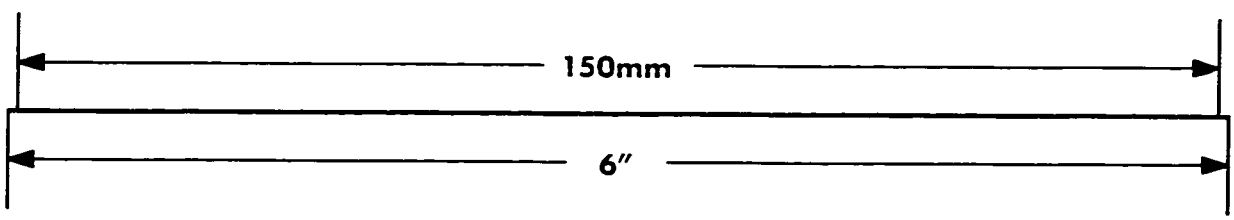
Chapter 9

Vita

- Hakim ADICHE
- Born in Algiers, Algeria on March 22, 1970.
- Received Bachelor's degree in Electrical Engineering option Computer Engineering from the National Institute of Electricity and Electronics (INELEC), Algeria in June , 1993.
- Completed Master's degree requirements at King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia in November, 1997.

IMAGE EVALUATION
TEST TARGET (QA-3)





APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989